VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

EKONOMICKÁ FAKULTA

KATEDRA APLIKOVANÉ INFORMATIKY

Vývoj servisní mobilní aplikace pro monitoring průmyslových praček

Development of Service Mobile Application for Industrial Laundry Machines Monitoring

Student: Bc. Aleš Miňo

Vedoucí diplomové práce: RNDr. Jaroslav Ševčík, Ph.D.

Ostrava 2017

VŠB - Technická univerzita Ostrava
Ekonomická fakulta
Katedra systémového inženýrství

Zadání diplomové práce

Student: **Bc. Aleš Miňo**

Studijní program: N6209 Systémové inženýrství a informatika

Studijní obor: 6209T025 Systémové inženýrství a informatika

Téma: Vývoj servisní mobilní aplikace pro monitoring průmyslových praček
Development of Service Mobile Application for Industrial Laundry Machines Monitoring

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Úvod
2. Teoretická východiska tvorby mobilních aplikací v oblasti průmyslových praček
3. Analýza současného stavu podniku v oblasti komunikace
4. Návrh a implementace vyvíjené aplikace
5. Závěr

Seznam použité literatury
Seznam zkratek
Prohlášení o využití výsledků diplomové práce
Seznam příloh
Přílohy

Seznam doporučené odborné literatury:

LACKO, Ľuboslav. *Vývoj aplikací pro Android.* Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.
OSHANA, Robert and Mark KRAELING. *Software engineering for embedded systems: methods, practical techniques, and applications.* Boston: Elsevier/Newnes, 2013. ISBN 978-0124159174.
SMITH, Dave. *Android recipes: a problem-solution approach for Android 5.0.* Fourth ed., New York: Apress, 2015. ISBN 978-1-484204-76.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **RNDr. Jaroslav Ševčík, Ph.D.**

Datum zadání: 18.11.2016
Datum odevzdání: 21.04.2017

doc. Ing. Jana Hančlová, CSc.
vedoucí katedry

prof. Dr. Ing. Zdeněk Zmeškal
děkan fakulty

Prohlášení

Prohlašuji, že jsem celou práci, včetně všech příloh, vypracoval samostatně.

Ve Frýdku - Místku dne 19. 4. 2017

.....

Bc. Aleš Miňo

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce RNDr. Jaroslavu Ševčíkovi Ph.D. za pomoc a cenné rady z oblasti vývoje softwarových aplikací. Dále Martinu Čechákovi za konzultace na téma vývoj mobilní aplikace pro platformu Android.

Obsah

1 Úvod

V současnosti se mobilní technologie využívají téměř ve všech oblastech moderního života a je prakticky nemyslitelné, aby moderní člověk nevlastnil alespoň jedno chytré zařízení. Mobilní zařízení se stala pomyslnou pravou rukou člověka, a to ne pouze pro běžné telefonování a posílání textových zpráv, jako tomu bylo kdysi. V tuto chvíli lze tyto zařízení využít k nejrůznějším účelům, jako je navigace prostřednictvím určení geografické polohy, interakce s okolním světem prostřednictvím sociálních sítí, fotografování, platebnímu styku a mnoho dalšího. To všechno lze provádět pomocí vytvořených aplikací.

Ani průmyslová sféra už není výjimkou a se stroji jako jsou průmyslové pračky lze komunikovat prostřednictvím mobilního zařízením. To opět usnadní řadu věcí a zpříjemní uživatelům jejich používání praček. Prostřednictvím aplikací lze také průmyslové pračky monitorovat, a díky tomu přehledně zobrazit důležité informace.

Tématem této práce je navrhnout, vytvořit a implementovat servisní mobilní aplikaci na platformě Android, sloužící především pracovníkům vývojového oddělení, kteří neustále přicházejí s inovacemi a posouvají tak průmyslové pračky na vyšší úroveň. Tato aplikace je vyvíjena pro společnost Alliance Laundry CE s.r.o.

Alliance Laundry CE s.r.o. se v současnosti řadí mezi špičkové výrobce průmyslové prádelenské techniky v Evropě s pevnou pozicí na celosvětovém trhu. Jedno ze svých hlavních sídel má právě ve městě Příbor. Toto sídlo disponuje špičkovým vývojovým oddělením, které je rozdělené na konstrukční a softwarovou oblast. Tato aplikace bude sloužit pro usnadnění vývoje na softwarovém oddělení. Prostřednictvím aplikace by mělo jít snáze testovat vyvíjený software do praček a identifikovat případné chybové stavy.

Mobilní aplikace by měla splňovat tyto základní funkce:

- navázání komunikace s pračkou prostřednictvím Bluetooth technologie,
- zobrazení servisních informací o připojeném zařízení,
- zobrazit podrobný přehled jednotlivých pracích cyklů, které na vybraném zařízení proběhly,
- poskytnout uživateli auditní informace o připojeném zařízení,
- disponovat jednoduchým grafickým designem s intuitivním ovládáním.

V druhé kapitole práce jsou popsána teoretická východiska pro tvorbu mobilních aplikací v oblasti průmyslových praček. Jsou zde uvedeny základní informace o průmyslových pračkách a jejich vestavěných systémech. Součásti kapitoly je také popis samotné platformy, pro kterou je mobilní aplikace vyvíjena. Na základě zde uvedených poznatků je prováděna aplikační část této práce.

Následující kapitola se zabývá analýzou současného stavu podniku v oblasti komunikace. V této kapitole je popsáno, jakým způsobem probíhá monitoring průmyslových praček v současné době včetně stručného popisu společnosti.

Čtvrtou kapitolou je samotný návrh a implementace aplikace. Tato kapitola se opírá o uvedené teoretické poznatky v druhé části této práce. V této kapitole je popsán vývoj aplikace zahrnující návrh, implementaci a následné nasazení do provozu.

Poslední kapitolou je závěr, který se zabývá celkovým zhodnocením práce a jednotlivých cílů včetně jejich dosažení.

2 Teoretická východiska tvorby mobilních aplikací v oblasti průmyslových praček

Cílem této kapitoly je poskytnout základní teoretickou podporu pro následný vývoj mobilní aplikace. V první části této kapitoly jsou popsány základy technologií, na kterých průmyslové pračky fungují včetně způsobu komunikace s mobilním zařízením. Ačkoliv se v praktické části tyto pojmy nevyužívají, je naprostou nezbytnosti poskytnout alespoň základní přehled informací o zařízení, pro které je mobilní aplikace vyvíjena.

V druhé části je popsán způsob vývoje mobilní aplikace, což zahrnuje využité metodiky a nástroje. V další části je popsán operační systém Android. Tento popis je uveden hlavně z pohledu vývojáře, kdy jsou uvedeny základní koncepty pro vývoj aplikací na této platformě. Samozřejmostí je uvedení i některých obecných informací. Poslední část zahrnuje stručný popis technologií, které byly při vývoji využity.

2.1 Technologie průmyslových praček a jejich vestavěných systémů

V první části této podkapitoly jsou popsány základy systémů, které pracují v reálném čase. Poté se prostřednictvím této kapitoly autor zaměří na již konkrétní popis vestavěných systémů, kterými průmyslové pračky bezpochyby jsou. Následuje stručná charakteristika průmyslových praček a způsobu jejích komunikace s mobilním zařízením, kterého je v této práci využito.

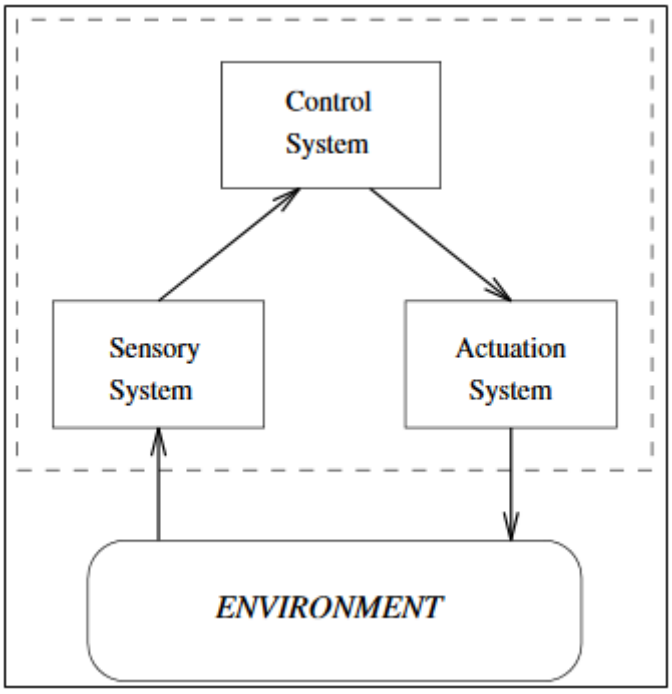
2.1.1 Systémy pracující v reálném čase

Real-Time systémy jsou výpočetní systémy, které musí reagovat v přesném časovém omezení na události, které v prostředí těchto systému nastanou. Důsledkem tohoto tvrzení je fakt, že správné chování těchto systémů nezáleží pouze na hodnotách výpočtů, ale také na čase, kdy jsou tyto výpočty prováděny. Reakce, která nastane příliš pozdě, může být zbytečná, nebo dokonce nebezpečná. V současné době hrají tyto systémy rozhodující roli v naší společnosti a jsou využívány v nejrůznějších oblastech. Příklady oblastí, které vyžadují výpočetní systémy pracující v reálném čase, jsou následující:

- systémy pro řízení letu letadel,

- telekomunikační systémy,
- zdravotnické systémy,
- průmyslové systémy,
- virtuální realita,
- multimediální systémy. (Buttazzo, 2011)

Základním předpokladem pro správné fungování systému pracujících v reálném čase je myšlenka, že okolní prostředí je vždy nejdůležitější komponentou jakéhokoliv systému pracujícího v reálném čase. Obrázek 2-1 ukazuje typické blokové schéma pro architekturu systému pracujícího v reálném čase. (Buttazzo, 2011)



Obrázek 2-1 Blokové schéma typického systému pracujícího v reálném čase, zdroj: Buttazo (2011, s. 8)

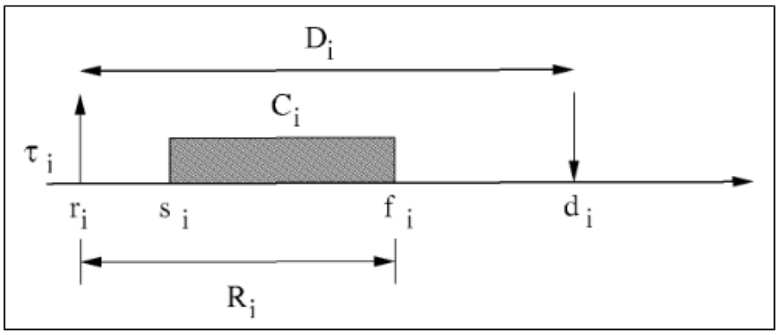
Časování systémů pracujících v reálném čase

Systém pracující v reálném čase je obvykle modelován jako sada souběžně probíhajících úloh. Každá úloha představuje výpočetní proces, který musí být proveden podle stanovených omezení. (Kopetz, 2011)

Nejvíce významné časovací parametry, které jsou obvykle definovány v každém výpočetním procesu systému pracujícího v reálném čase, jsou uvedeny zde:

- **Doba uvolnění** r_i: jedná se o dobu, při které je daná úloha připravena k provedení. Často bývá označována jako doba příchodu.
- **Doba spuštění** s_i: jedná se o dobu, při které je daná úloha spuštěna poprvé.
- **Výpočetní doba** C_i: čas, který je nezbytně nutný pro provedení výpočtu procesorem bez jakéhokoliv přerušení.
- **Doba dokončení** f_i: jedná se o čas, kdy je dokončeno provedení úlohy.
- **Doba odpovědi** C_i: představuje čas, který uběhne od doby uvolnění úlohy, a jejím dokončení.
- **Absolutní doba ukončení (deadline)** d_i: jedná se o čas, před kterým by měla být daná úloha dokončena.
- **Relativní doba ukončení (deadline)** D_i: je čas, který je relativní k době uvolnění. Představuje tak dobu mezi absolutním ukončením a dobou uvolnění, kdy by měla být daná úloha dokončena. (Buttazzo, 2005)

Použití těchto parametrů je schematicky vyjádřeno na obrázku 2-2, kde doba uvolnění je reprezentována šipkou nahoru a doba absolutního ukončení pak šipkou dolů.



Obrázek 2-2 Využití typických časovacích parametrů při úloze v reálném čase, zdroj: Buttazzo (2005, s. 3)

2.1.2 Charakteristika zařízení s vestavěným systémem

Vestavěný systém je jednoúčelový systém, ve kterém je řídící jednotka zabudována přímo do zařízení, které ovládá. Tyto vestavěné systémy se odlišují od normálních počítačů svou jednoúčelovostí a jsou specializované pro předem definované činnosti. (Douglass, 1999)

Vestavěný systém, který pracuje v reálném čase, je vždy součástí většího systému, který lze nazvat jako inteligentní produkt. Tento inteligentní produkt obsahuje fyzický subsystém, jehož prostřednictvím je ovládán vestavěný systém. Tento systém se nazývá nejčastěji uživatelské rozhraní. (Pohl et al., 2016)

Stále rostoucí poměr výkon / cena mikroprocesorů činí ekonomicky atraktivní nahradit konvenční mechanické nebo elektronické kontrolní systémy, které se využívají v mnoha produktech vestavěnými systémy pracujícími v reálném čase. Existuje mnoho příkladů produktů, které využívají vestavěné systémy. Těmito produkty jsou například:

- mobilní telefony,
- kontrolní systémy v automobilu,
- zdravotnické přístroje,
- tiskárny,
- televize,
- průmyslové pračky. (Pohl et al., 2016)

Trh s vestavěnými zařízení je charakterizován neustálým tlakem na inovace a všude přítomnou potřebou, co nejvíce redukovat náklady na vývoj. Tyto znaky jsou doprovázeny nutností vyvíjet nové inovativní produkty s větší funkcionalitou, které mohou být zákazníkovi nabídnuty. Výsledkem výše uvedených požadavků je, že vyvíjené vestavěné systémy jsou více komplexní a v současné době také více využívány. (Pohl et al., 2016)

2.1.3 Průmyslové pračky

Jedná se o specifický druh praček, který nabízí rozšířené možnosti v prádelenské technice. Tyto pračky jsou uzpůsobené pro praní velkého objemu prádla. Charakteristickým rysem je optimalizace energetické spotřeby a úspora času. Disponuje displejem, který představuje uživatelské rozhraní. Nejčastěji se využívají

v hotelech, nemocnicích a prádelnách. V příloze 1 lze vidět konkrétní ukázku průmyslové pračky.

Průmyslová pračka jako zařízení s vestavěným systémem

Jako příklad vestavěného systému lze uvést průmyslovou pračku. V hlavním ovladači průmyslové pračky se nachází mikroprocesor, který vytváří jádro systému a představuje tak základní hardwarovou část nejnižší vrstvy architektury. Jednotlivé softwarové vrstvy, které s tímto mikroprocesorem pracují a přidávají různé funkcionality hlavnímu ovladači. Konkrétní softwarové funkce jsou potom volány na základě uživatelova vstupu, zadaného nejčastěji prostřednictvím klávesnice, která představuje uživatelské rozhraní v podobě hardwaru a pracuje nad všemi softwarovými vrstvami. Charakteristickým znakem pro tyto zařízení je zapouzdření jednotlivých softwarových a hardwarových vrstev, kdy softwarové vrstvy jsou vestavěny mezi vrstvy hardwaru. (Oshana a Kralieng, 2013)

Základní popis řídícího systému průmyslových praček

Tento řídící systém je využíván u praček, pro které je monitorovací mobilní aplikace vyvíjena a jednoduše tak ilustruje možnosti, kterými průmyslová pračka disponuje. Výčet funkcí řídícího systému je uveden v dalším textu.

Řídící systém má tedy řadu funkcí, mezi které například patří:

- 99 detailních programovatelných programů pro provedení pracích cyklů,
- regulaci vnějších čerpadel nebo tekutých pracích prostředků,
- rozložení prádla tak, aby bylo zajištěno rovnoměrné vyvážení,
- automatické vyvážení teploty během napouštění,
- nastavení možností a konfigurací,
- lokalizovatelnost pro jednotlivé země.

Při běhu programu se pak uživateli zobrazí následující údaje:

- zvolený program,
- aktivní krok praní,
- zbývající čas v rámci programu,
- ukazatel průběhu pracího cyklu,
- diagnostická hlášení.

Řídící systém disponuje provozní nabídkou, která umožňuje:

- program ručně zkrátit, prodloužit nebo zastavit,
- naprogramovat pauzu,
- přímé ovládání zvolených prvků (vodní ventily apod.).
- přehled programů,
- servisní informace.

Hardware a software modulu vestavěného systému pracího stroje disponuje následujícím:

- snadným ovládáním pomocí srozumitelné klávesnice,
- hardwarem obsahujícím dvě elektronické desky reprezentující uživatelské rozhraní,
- řídicím softwarem pracího stroje uloženým v interní paměti programátoru a snadno upravovatelným prostřednictvím USB (Universal Serial Bus) připojení,
- pracími programy uloženými v paměti nezávislé na napájení.

2.1.4 Komunikace průmyslových praček s mobilním zařízením

Pro komunikaci mezi průmyslovými pračkami a mobilním zařízením je využita technologie Bluetooth. Jako zprostředkovatel komunikace funguje Bluetooth senzor, který je integrovaný v pračce. Konkrétní technologie komunikace, která je využívaná se nazývá „*Bluetooth Low Energy*“, dále jen BLE.

Bluetooth

Bluetooth je bezdrátový technologický standard, který slouží pro výměnu dat na krátkou vzdálenost mezi dvěma zařízeními. Tato technologie pracuje v síti, která je označována jako PAN (Personal Area Network), což v překladu znamená osobní síť. Jedná se o počítačovou síť, která je typicky tvořena zařízeními, jako jsou mobilní telefony. Dosah této sítě je velmi malý, řádově se jedná o několik metrů. Bluetooth je standardizováno jako IEEE 802.15.1 a jedná se o skupinu, která je založena na technologii Bluetooth. (Bluetooth SIG, Inc., c2017)

Bluetooth zařízení využívá rádiové vlny namísto drátů nebo kabeláže pro připojení například k telefonu či počítači. Rádiové vlny pracují na frekvenci ultra krátkých vln, kdy rozsah této frekvence je stanoven přibližně od 300 MHz do 3GHz. Každý Bluetooth produkt obsahuje malý počítačový čip s Bluetooth rádiem a softwarem, který toto připojení usnadňuje. Moderní komunikace mezi Bluetooth zařízeními probíhá na krátkou vzdálenost s využitím Ad-hoc sítě (k tomuto účelu vytvořené), známe jako „*piconet*“. Jedná se o síť, kde jedno zařízení funguje jako server a až sedm jiných zařízení se k němu může připojit. (Bluetooth SIG, Inc., c2017)

Bluetooth Low Energy

Bluetooth povoluje vytváření malých senzorů. S příchodem nové funkcionality BLE jsou vývojáři schopni vytvořit malé senzory, které jsou napájeny malými „*knoflíkovými“* bateriemi po dobu několika měsíců, v některých případech i let. Mnohé z těchto senzorů využívají tak malé množství energie, že vývojáři našli způsoby, jak tuto energii v průběhu dobíjet, což zajištuje senzorům potenciálně neomezenou životnost. Využití této technologie lze v dnešní době pozorovat u více než miliard zařízení. (Bluetooth SIG, Inc., c2017)

BLE je založeno na zcela novém frameworku, který využívá generické atributy (GATT). Z vývojářského pohledu je GATT extrémně flexibilní a může být využít pro téměř jakýkoliv scénář. Výsledkem je, že Bluetooth pouze nepropojí dvě zařízení s cílem obyčejného přenosu dat v extrémně úsporném režimu, ale přímo umožnuje propojit zařízení s aplikací, která běží na mobilním telefonu, tabletu nebo počítači. Prostřednictvím této aplikace pak s připojeným zařízením manipulovat. Extrémně nízká energetická náročnost a využití GATT je v současné době hlavním důvodem rozmachu internetu věcí. (Bluetooth SIG, Inc., c2017)

BLE Android aplikační programové rozhraní

Vestavěná podpora pro BLE byla zavedena poprvé ve verzi Android 4.3 (viz kapitola 2.3). Poskytuje aplikační rozhraní (API), které může aplikace využít k vyhledávání dostupných zařízení, dotazování na jejich služby a přenosu informací. (Android Developers, c2017a)

Na rozdíl od klasického Bluetooth, BLE je navrženo tak, aby zajistilo co nejmenší spotřebu energie. To povoluje aplikacím běžícím na platformě Android komunikovat s BLE zařízeními, které mají striktně stanovené požadavky na spotřebu energie. Jako příklad lze uvést Bluetooth senzory nebo monitory pro kontrolu srdeční funkce. (Android Developers, c2017a)

Mezi klíčové pojmy, které charakterizují BLE patří:

- **Generický atribut (GATT**) – představuje všeobecnou specifikaci pro zasílání a přijímání malých částí dat, známé jako „*atributy“*.
- **Protokol atributů (ATT) –** GATT je postaven nad tímto protokolem, což má často za následek, že jsou tyto pojmy zaměňovány. Tento protokol je optimalizován pro běh na BLE zařízeních a definuje pravidla pro komunikaci. Každý atribut je identifikován prostřednictvím univerzálního unikátního identifikátoru (UUID), který je standardizován jako 128 bitový formát. Atributy, které jsou přenášeny pomocí ATT jsou formátovány jako charakteristiky a služby.
- **Charakteristika** – Obsahuje jednu hodnotu a libovolný počet deskriptorů, které tuto charakteristickou hodnotu popisují. Charakteristickou hodnotu si lze představit jako datový typ, který je analogický k třídě.
- **Deskriptor** – Deskriptory jsou definované atributy, které popisují charakteristickou hodnotu. Slouží k vytvoření popisu, který je čitelný pro člověka. Například se může jednat o měřitelnou jednotku, která je specifická pro danou charakteristiku.
- **Služba** – Jedná se o kolekci charakteristik. Například může existovat služba pod názvem „*Monitor srdeční funkce*“, která zahrnuje charakteristiky, jako je například měření tepové frekvence. (Android Developers, c2017a)

2.2 Nástroje a metodiky vývoje mobilní aplikace

V této kapitole je popsána metodika Unified Software Development Process a její grafický nástroj UML (Unified Model Language). V rámci této kapitoly jsou popsány i základní koncepty agilního vývoje a využití sady technik uživatelského prožitku, známe pod názvem User Experience.

2.2.1 Unified Software Development Process

Jedná se o iterativní metodiku, která vznikla v roce 1999. Jelikož je název metodiky poněkud krkolomný, tak se v praxi běžně využívá označení Unified Process, případně zkratka UP. Označení UP je využíváno i v další části práce. Kořeny této metodiky však sahají až do roku 1987, kdy s touto myšlenkou přišel Ivar Jacobson, který je považovaný za zakladatele. (Kadlec, 2004)

UP je standardem pro vývoj softwaru. Tato metodika je úzce propojena s modelovacím jazykem UML. Pro využití této metodiky platí, že pro každý projekt je třeba vytvořit novou instanci (konkrétní podobu) této metodiky. Samotní autoři respektují zásady, že každý projekt je jiný a nelze použít jednu metodiku pro vývoj jakéhokoliv systému či aplikace. Proto je pro vývojáře nutné si z jednotlivých metodik vybrat to, co je pro daný projekt nejvhodnější, a tak stanovit optimální způsob vývoje. (Kadlec, 2004)

Základní myšlenky metodiky UP

Základními myšlenkami a axiomy, na kterých je tato metodika postavena jsou podle Arlow a Neustadt (2007, s. 59):

- zásada řízení případem užití a rizikem,
- zásada soustředění se na architekturu,
- zásada iterace a přírůstku.

U metodiky UP platí, že při návrhu i vývoji je klíčovým fragmentem případ užití, tedy textový popis využití systému či aplikace jeho konkrétním uživatelem. Pomocí případů užití jsou tedy zachyceny konkrétní uživatelské požadavky a na jejichž základě je stanoven optimální postup vývoje. (Kadlec, 2004)

Rizika a jejich analýza jsou také součásti této metodiky. Rizika se nacházejí všude a je třeba se na ně připravit a aktivně je vyhledávat v průběhu vývoje. Tímto postupem lze tak dosáhnout jejich eliminace již v počátku vzniku. (Arlow a Neustadt, 2007)

Metodika UP se řídí pravidlem, že kvalitní software může jen těžko vzniknout na základě špatně navržené architektury. Důležitou složkou procesu je tedy pečlivý

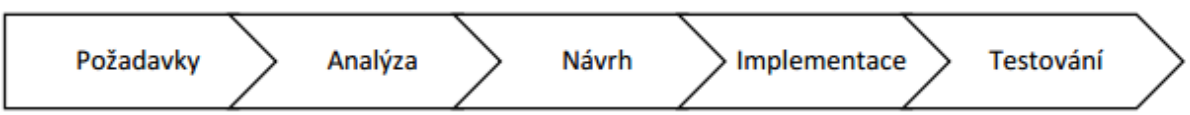
návrh architektury, který zaručí požadovanou kvalitu softwaru. (Arlow a Neustadt, 2007)

Vývoj prostřednictvím této metodiky probíhá iterativně a inkrementálně. Dodání požadovaného softwaru tedy nenastane jednorázově, ale zákazníkovi jsou postupně dodávány nové funkce v rámci vyvíjeného softwaru. Prostřednictvím iterací je tedy projekt rozdělen do menších lépe zvládnutelných části. Při využívání iterativního vývoje není plánování akcí tak složité, jelikož objem prací pro jednu iteraci je menší než v případě celého projektu (vodopádový způsob vývoje). Lze tak lépe naplánovat jednotlivé činnosti v rámci vývoje softwaru. (Arlow a Neustadt, 2007)

Pracovní postupy metodiky UP

Jednotlivé iterace obsahují pět základních pracovních postupů, které určují, co je třeba udělat včetně toho, jakým způsobem toho lze dosáhnout. Níže uvedené pracovní postupy (viz obrázek 2-3) jsou pouze základní a přímo je definuje metodika UP. Jednotlivé iterace mohou obsahovat řadu dalších pracovních postupů, které mohou být pro různé iterace specifické, avšak ty již nejsou definovány v rámci metodiky UP. Zde je uvedeno pět základních pracovních postupu metodiky UP:

1. **Stanovení požadavků** při vývoji softwaru je klíčové. Správná formulace požadavků není triviálním úkolem, jelikož je zákazník většinou definuje vágně a nejasně. Cílem je tyto požadavky převést do jednotlivých případů užití.
2. **Analýza** je klíčovým prvkem pro specifikaci získaných požadavků, jejich strukturování nebo ohodnocení podle důležitostí.
3. **Návrh** slouží k co nejpřesnějšímu zachycení všech požadavků a jejich následné implementaci do architektury systému.
4. **Implementace** již představuje samotné psaní programového kódu jednotlivých modulů.
5. **Testování** slouží k ověření správné funkcionality implementovaných modulů. (Kadlec, 2004)



Obrázek 2-3 Pracovní postupy, zdroj: Čechák (2015, s. 26)

Kadlec (2004, s. 108) tvrdí, že: „*Délka jednotlivých iterací je proměnlivá a pružná, avšak obecně platí, že žádná iterace v metodice UP by neměla trvat déle než dva až tři měsíce.*“

Fáze metodiky UP

Životní cyklus projektu je v metodice UP rozdělen na čtyři fáze, které lze vidět na obrázku 2-4. Jednotlivé fáze jsou popsány v následujících odstavcích z pohledu jejích cílů a probíhajících činností. Každá fáze se může skládat z několika iterací. V jednotlivých iteracích jsou pak realizovány základní pracovní postupy.

Obrázek 2-4 Fáze metodiky UP, zdroj: Čechák (2015, s. 26)

Fáze zahájení, v jejímž rámci jsou vytvářeny podmínky proveditelnosti. Provádí se zachycení podstatných požadavků a důležitých rizik. Cílem této fáze je dosažení jednotné shody, jakým směrem se bude projekt ubírat napříč všemi zúčastněnými stranami. (Kadlec, 2004)

Fáze rozpracování, jejímž cílem je vytvoření stabilního architektonického základu, vylepšení odhadů rizik a stanovení kvalitativních parametrů. Součástí této fáze je i zachycení již většiny případů použití a tvorba detailního plánu konstrukce. (Arlow a Neustadt, 2007)

Fáze konstrukce, jejímž úkolem je postupné zpřesňování požadavků a vytváření výsledné aplikace. Důležité je klást důraz na zachování integrity již vytvořené architektury. (Kadlec, 2004) (Arlow a Neustadt, 2007)

Fáze zavedení, kdy nastává předání produktu zákazníkovi včetně veškeré dokumentace. Tato fáze zahrnuje také následnou podporu například v rámci školení a údržby až do okamžiku, kdy je zákazník spokojen. (Kadlec, 2004)

2.2.2 Unified Modelling Language

Jedná se o v překladu unifikovaný modelovací jazyk, který je v praxi označován zkratkou UML. UML představuje standardizovaný grafický nástroj, který je využíván při vývoji softwaru pomoci metodik UP a RUP. Cílem UML je graficky znázornit strukturální a behaviorální složku vyvíjeného softwaru. Tato grafická notace

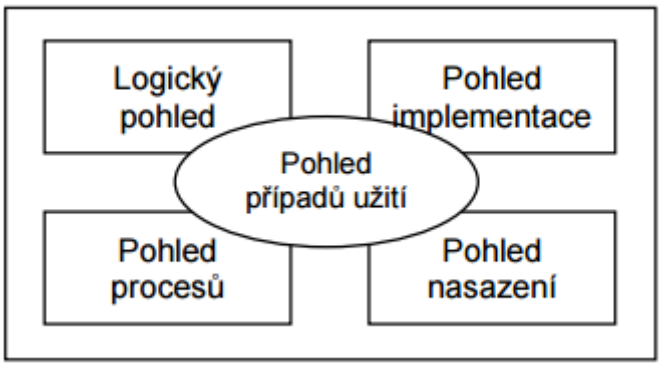
potom slouží jako prostředek komunikace ve vývojovém týmu, a také se všemi dalšími zúčastněnými stranami. (Hamilton a Miles, 2006) (Čechák, 2015)

Objektově orientovaná analýza systému

UML se nejčastěji využívá pro grafickou anotaci objektově orientované analýzy. V rámci objektově orientované analýzy jsou zkoumány třídy, což jsou speciální datové typy, umožňující deklaraci datových položek a funkcionalit jako jsou metody, funkce a procedury. Instance tříd se nazývají objekty. (Hamilton a Miles, 2006) (Čechák, 2015)

Model pohledů

Je mnoho způsobů, jak popsat model diagramů, aby bylo možné zachytit hlavní aspekty vyvíjeného softwaru. Jeden z nejznámějších způsobů je model pohledů Phillipa Kruchtena 4+1, který zařazuje využití UML nástrojů do pěti navzájem se prolínajících skupin z pěti různých pohledů (Obrázek 2-5). Jednotlivé pohledy jsou stručně popsány v následujících odstavcích. (Hamilton a Miles, 2006) (Čechák, 2015)



Obrázek 2-5 Diagram 4+1 pohledů, zdroj: Čechák (2015, s. 27)

Logický pohled popisuje abstraktní strukturu jednotlivých části vyvíjeného softwaru. Zahrnuje diagramy tříd, objektů, stavů a interakcí. (Hamilton a Miles, 2006)

Pohled procesů popisuje procesy, které probíhají v systému. Tento pohled typicky zahrnuje diagramy aktivit. (Hamilton a Miles, 2006)

Pohled implementace slouží k popisu architektury. Popisuje rozdělení systému do jednotlivých modulů a komponent. Pro tento pohled je typický diagram komponent a balíčků. (Hamilton a Miles, 2006)

Pohled nasazení popisuje fyzické nasazení systému a pro tento pohled se využívá diagram nasazení. (Hamilton a Miles, 2006)

Pohled případů užití popisuje funkcionalitu systému z pohledu okolního světa. Tenhle pohled je nutný ke stanovení, co musí daný systém splňovat. Tento pohled je základem pro všechny ostatní pohledy, a proto se také využívá označení 4+1. (Hamilton a Miles, 2006) (Čechák, 2015)

UML nabízí k využití více než 15 diagramů. Konkrétní diagramy, které jsou v práci uvedeny, jsou popsány až v aplikační části.

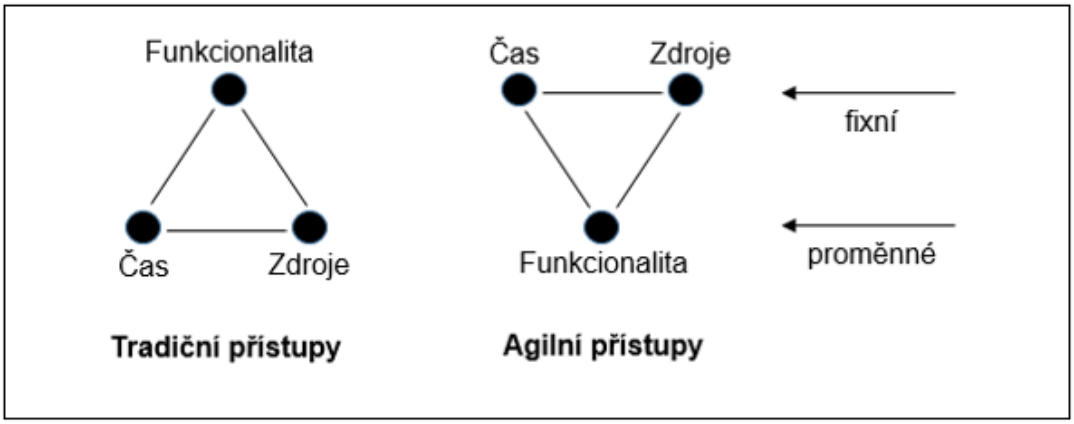
2.2.3 Agilní vývoj

Metodiky, které umožňují co nejrychlejší vývoj softwaru, jeho průběžnou údržbu a rychlou reakci na měnící se požadavky se nazývají agilní. Vznik těchto metodik lze zařadit zhruba na přelomu let 2000 a 2001. (Kadlec, 2004)

Pojem agilní metodika lze definovat jako skupinu metodik, která vychází z poznání, že jediným možným způsobem, jak prověřit správnost navrženého systému, je co možná nejrychleji jej vyvinout, předložit zákazníkovi a na základě zpětné vazby upravovat. (Kadlec, 2004)

Základní principy agilních metodologií

Rozdíl mezi tradičním způsobem vývoje, kdy zákazník většinou na začátku stanoví požadavky, na kterých je postavena celá funkcionalita a agilním přístupem lze nejlépe pozorovat na obrázku 2-6. (Kadlec, 2004)



Obrázek 2-6 Rozdíl mezi tradičními a agilními programovacími přístupy, zdroj: Kadlec (2004, s. 119)

Tradiční metodiky se zakládají na nutnosti naplnit za každou cenu dokument, který lze označit jako specifikaci požadavků. Jednotlivé požadavky – tedy funkcionalita jsou fixní, zatímco v roli proměnných vystupují čas a potřebné zdroje. Při tradičním vývoji je množina požadavků neměnná a je stanovena na počátku vývojového procesu. Hlavním cílem je naplnění těchto požadavků. V roli proměnných vystupuje čas a zdroje, jelikož je dopředu známo, jaké funkce by měl daný vyvíjený software mít, avšak nelze určit, jak dlouho bude software vyvíjen a kolik peněz to bude stát. (Kadlec, 2004)

Agilní přístupy fungují naprosto obráceně. Čas a zdroje jsou považovány za fixní, jinými slovy stanovenými na začátku vývoje (zadavatelem). Na začátku projektu je přesně stanoven nejdelší možný čas vývoje a nejvyšší možné náklady. Jediné, co se v průběhu vývoje mění je funkcionalita, která je přizpůsobována flexibilně novým požadavkům. Základní principy agilních metodik jsou popsány v následujících odstavcích. (Kadlec, 2004)

Iterativní a inkrementální vývoj s velmi krátkými iteracemi patří mezi základní principy agilního vývoje. Plán projektu je navržen tak, že nové funkcionality se dodávají zákazníkovi často, v některých případech i denně. Zákazník je průběžně konfrontován s novými konfiguracemi vyvíjeného softwaru. Hlavní výhodou je fakt, že zákazník je neustále součástí vývoje a má neustálý přehled o tom, v jaké fázi se vývoj nachází a na čem vývojový tým aktuálně pracuje. Z toho vyplývá, že na konci vývoje obdrží zákazník opravdu požadovaný produkt, který splňuje požadavky stanovené na začátku a v průběhu vývoje. (Kadlec, 2004) (Stober a Hansmann, 2010)

Důraz na přímou, osobní komunikaci v týmu patří mezi další z principů základního vývoje. Nejčastějším problémem při práci ve skupinách a jiných sociálních interakcí je komunikace. Agilní přístupy zařazují komunikaci jako nedílnou součást vývojové procesu. To zahrnuje časté schůzky, skupinové programování atd. Tento proces má za důsledek dřívější odhalení potenciálních chyb a rizik, které v průběhu mohou nastat. (Kadlec, 2004) (Stober a Hansmann, 2010)

Neustálý kontakt se zákazníkem představuje jeden z hlavních pilířů agilního vývoje. V ideálním případě by měl být zákazník členem vývojové týmu a spolupodílet se tak na celkovém návrhu a vývoji softwaru. (Kadlec, 2004)

Průběžné a opakované provádění testů je vzhledem k častým změnám, které při agilním vývoji nastávají nezbytnou součástí vývojového procesu. Pravidelně je tedy nutné ověřovat správnost navrženého softwaru. Testy by měly být automatizované a jejich správná a včasná implementace zaručuje v konečném důsledku požadovanou kvalitu softwaru. (Kadlec, 2004) (Stober a Hansmann, 2010)

Agilní metodiky vyžadují mnohem menší množství formálních a byrokratických dokumentů. Základní přesvědčení tohoto typu vývoje je, že hlavním nositelem informací je samotný programový kód, na jehož formu je kladen největší důraz. (Kadlec, 2004)

2.2.4 User Experience

User Experience (UX) lze definovat jako sadu technik a metod, které se využívají pro návrh konkrétního uživatelského rozhraní. Překlad tohoto anglického výrazu zní „*uživatelský prožitek*“. V současné době se UX rozděluje do několika disciplín, kterými jsou:

- uživatelský výzkum (rozhovory, online průzkumy atd.),
- interakční design (zabývá se návrhem konkrétních stránek nebo interakcí složitých aplikací),
- informační architektura (definice struktury informací),
- vizuální design (návrh vzhledu aplikací, kontrastů barev a celkového působení). (Norman a Nielsen, c1998-2017)

UX zahrnuje všechny aspekty interakce koncového uživatele s firmou, jejími službami a výsledným produktem. Cílem UX je vytvořit produkt, který přesně splňuje požadavky a potřeby zákazníka. Dále je nutné dbát na to, aby neměl koncový uživatel pocit, že je neustále obtěžován a jeho celkový uživatelský prožitek byl tak maximalizován. (Norman a Nielsen, c1998-2017)

2.3 Mobilní operační systém Android

V této kapitole je popsán mobilní operační systém Android. Jedná se o platformu, pro kterou je servisní mobilní aplikace vyvíjena. Cílem této kapitoly je čtenáře stručně obeznámit s cílovou platformou.

2.3.1 Základní charakteristika

Android je operační systém vyvíjený společností Google. Tento systém je založen na jádře operačního systému Linux. V současnosti se využívá v mobilních telefonech, tabletech, chytrých hodinkách, brýlích a mnoha dalších chytrých zařízení. Jedná se o open-source projekt, jinými slovy může tento systém kdokoliv zdarma použít a upravit pro své konkrétní potřeby. Uživatelé mají k dispozici zdrojové kódy, které mohou využívat. (Android Developers, c2017b)

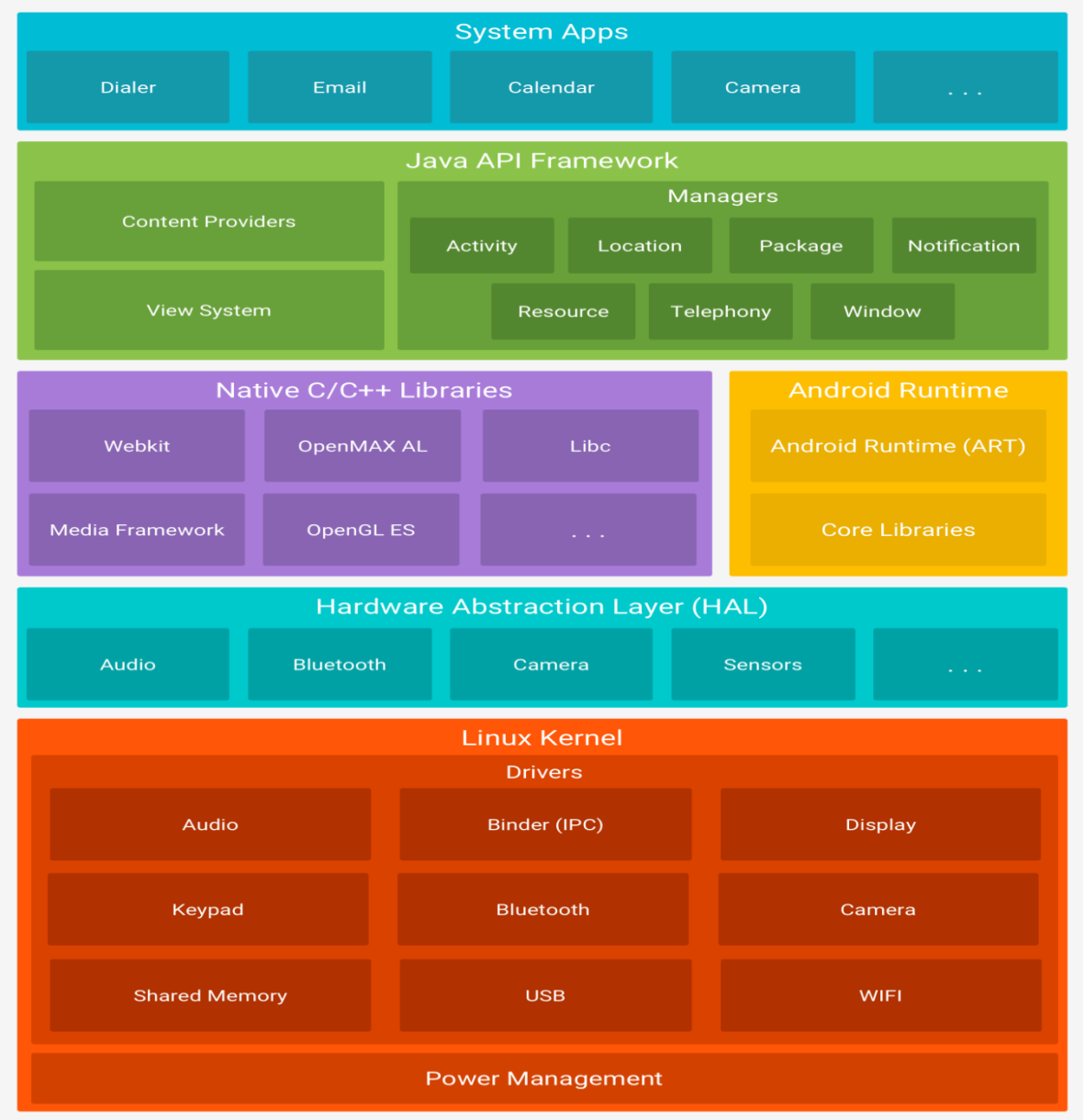
V dnešní době běží operační systém Android na více než miliardě mobilních zařízení a je využíván ve více než 190 zemích světa. Jedná se o nejrozšířenější mobilní platformu, jejichž využívanost každým dnem roste. (Android Developers, c2017b)

Softwarové a hardwarové hranice tohoto systému jsou neustále posouvány dopředu, tak aby přinesli uživatelům a vývojářům nové možnosti. Pro vývojáře je extrémně rychlý vývoj tohoto operačního systému možností, jak vytvářet stálé nové aplikace, které jsou více komplexní a využitelné. (Android Developers, c2017b)

Android poskytuje všechno, co vývojář potřebuje k vytvoření prvotřídní mobilní aplikace. Nabízí jednotný model, který umožňuje vyvíjenou aplikaci distribuovat stovkám miliónů uživatelů z celého spektra zařízení. Také poskytuje nástroje pro vytváření mobilních aplikací, z nichž nejznámější je Android Studio. (Android Developers, c2017b)

2.3.2 Architektura

Operační systém Android je otevřený systém (open-source), který je založený na linuxovém jádře, které je přizpůsobeno k využití široké škály zařízení. Na obrázku 2-7 lze vidět hlavní komponenty mobilní platformy Android. Jednotlivé komponenty jsou pak popsány v následujících odstavcích.



Obrázek 2-7 Architektura platformy Android, zdroj: Android Developers (c2017c)

Linuxové jádro (kernel) představuje základ mobilní platformy Android a jedná se o nejnižší vrstvu architektury. Je využíváno celé řady vlastností operačního systému Linux. Jako příklad lze uvést přidělování systémových prostředků (operační paměť, procesor). Hlavním důvodem pro využití jádra operačního systému Linux je jeho snadná přenositelnost mezi jednotlivými zařízeními. (Android Developers, c2017c)

Hardwarová abstraktní vrstva (HAL) disponuje standardním rozhraním, které umožňuje frameworku Java API přístup k možnostem hardwaru zařízení. Tato vrstva obsahuje vícenásobný počet modulů v podobě knihoven. Každá

z jednotlivých knihoven poté implementuje rozhraní, které je specifické pro danou hardwarovou komponentu. Jako příklad lze uvést kameru nebo Bluetooth modul. Jakmile je ze strany frameworku Java API požádáno o přístup k vybranému hardwarovému zařízení, tak je využita operačním systémem Android knihovna, která toto rozhraní implementuje. (Android Developers, c2017c)

Android runtime (ART) představuje běhové prostředí. Pro zařízení na kterých běží operační systém Android ve verzi 5.0 nebo vyšší je pro každou aplikaci dedikována jedna instance virtuálního stroje. ART je navrženo tak, aby i na nízko paměťových zařízeních mohlo běžet více instancí virtuálních strojů. Výchozím běhovým prostředím verze Android 4.4 bylo běhové prostředí Dalvik, ale uživatelé mohli zvolit i ART. V dřívějších verzích bylo k dispozici pouze běhové prostředí Dalvik. (Smith, 2015)

Velké množství komponent a služeb operačního systému Android, jako například ART a HAL, jsou založeny na nativních kódech, které vyžadují **nativní C/C++ knihovny**. Prostřednictvím platformy Android lze využít funkcionalitu těchto knihoven v rámci frameworku Java API. (Android Developers, c2017c)

Veškerá funkcionalita operačního systému Android je k dispozici prostřednictvím **Java API Framework.** Toto API tvoří základnu, která je nutná pro vývoj mobilních aplikací na platformě Android, a to díky znovu použitelným způsobům využití jádra, systémových komponent a služeb, které zahrnují:

- **View System**, který je vysoce využitelný při vytváření uživatelského rozhraní pro mobilní aplikace a zahrnuje například textová pole, tlačítka, seznamy nebo zaškrtávací políčka.
- **Resource Manager** zajišťující přístup k „*neprogramovým*“ zdrojům, kterými jsou například lokalizovatelné textové řetězce nebo grafické prvky.
- **Notification Manager,** který povoluje aplikacím zobrazit vlastní upozornění ve stavovém řádku.
- **Activity Manager,** který řídí životní cyklus aplikace a umožnuje tak přechody mezi jednotlivými aktivitami.
- **Content Provider,** který povoluje aplikacím přístup k údajům z jiných aplikací. (Android Developers, c2017c)

Na vrcholu architektury se nacházejí **systémové aplikace**. Jedná se o aplikace, které využívají běžní uživatelé a jsou předem předinstalované. Vyjmenovat lze například aplikaci pro zasílání SMS zpráv, pro správu systémového nastavení nebo odesílání emailů. Tyto aplikace hrají důležitou roli pro vývojáře. Pokud vyvíjí aplikaci, která vyžaduje funkcionalitu zasílání zpráv, nemusí jí nutně implementovat, nýbrž stačí využit již existující aplikace. (Android Developers, c2017c)

2.3.3 Základní aplikační komponenty

Aplikační komponenty jsou základními stavebními kameny pro vývoj mobilní aplikace na platformě Android. Každá komponenta představuje vstupní bod, prostřednictvím kterého může systém nebo uživatel pracovat s mobilní aplikací. Funkcionalita některých komponent může být závislá na jiných komponentách. (Android Developers, c2017d)

Aktivity

Aktivita představuje vstupní bod pro interakci s uživatelem. Reprezentuje jednu obrazovku s uživatelským rozhraním. Jako příklad lze uvést emailovou aplikaci. Tato aplikace má jednu aktivitu, která zobrazuje informace o doručené poště. Druhá aktivita může zobrazovat informace o odeslané poště. Ačkoliv jednotlivé aktivity pracují společně, aby vytvořili komplexní aplikaci, tak každá z aktivit je nezávislá na ostatních. (Android Developers, c2017d)

Služby

Služba je aplikační komponenta, která běží na pozadí mobilní aplikace a není součástí uživatelského rozhraní. Prostřednictvím služeb lze provádět dlouho trvající operace a řídit například síťové transakce nebo přehrávání hudby. Služba může být zavolána a spuštěna jinou programovou komponentou, například aktivitou. (Smith, 2015)

Příjemce vysílání

Aplikace mohou zasílat a přijímat vysílací zprávy ze systému Android a jiných Android aplikací. Tyto zprávy jsou vysílány, jakmile nastanou určité události. Může se jednat například o situaci, že Android systém odešle zprávu, jakmile nastane

systémová událost. Tuto systémovou událost si lze představit jako připojení mobilního zařízení do nabíječky. Aplikace si mezi sebou mohou také vyměňovat tzv. vysílací zprávy, například informace o stažení určitého obsahu apod. (Android Developers, c2017d)

Poskytovatelé obsahu

Jedná se o aplikační rozhraní, které umožňuje spravovat přístup k datům uložených v rámci samotné aplikace nebo jiných aplikacích. Také slouží pro sdílení dat mezi jednotlivými aplikacemi. Data jsou zapouzdřována a jsou definovány mechanismy pro zajištění jejich bezpečnosti. Poskytovatelé obsahu nabízejí granulární kontrolu nad oprávněním pro přístup k datům. Lze detailně nastavit, kdo a za jakých podmínek bude moct k datům přistupovat a využívat je. (Smith, 2015)

Záměry

Záměry představují zasílání zpráv, které lze interpretovat jako žádost o akci pro jiné aplikační komponenty. Prostřednictvím záměrů lze tedy řídit komunikaci mezi jednotlivými komponentami. Existují tři základní způsoby využití záměrů:

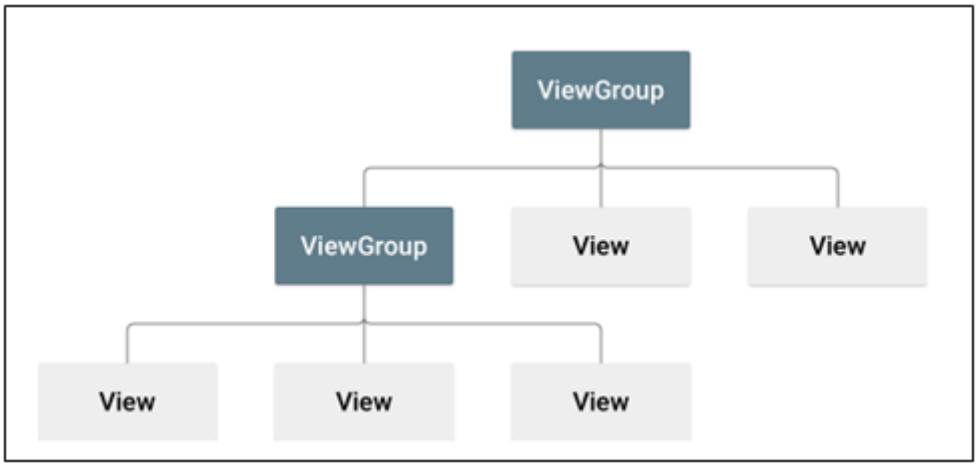
- **spuštění aktivity** – prostřednictvím záměru lze přecházet mezi jednotlivými aktivitami a předávat si důležitá data,
- **spuštění služby** – prostřednictvím záměru lze spustit služby, které běží na pozadí aplikace,
- **doručení vysílacích zpráv** – zasílání těchto zpráv lze provádět prostřednictvím záměrů. (Android Developers, c2017e)

Záměry se dále dělí na dva základní typy, kterými jsou **explicitní** a **implicitní.** Explicitní záměry přesně specifikují komponentu, která má být využita. Typicky se jedná o přechod mezi jednotlivými aktivitami v rámci aplikace. Implicitní záměry nedeklarují přesnou komponentu, ale pouze akci, která má být provedena. V mobilním zařízení je tato akce povolena k převzetí komponentou z jiné aplikace, která je pro to nejvíce vhodná. Může se jednat o přechod na internetovou stránku. Záměr je poté převzat webovým prohlížečem, který je v mobilním telefonu k dispozici. (Android Developers, c2017e)

Pohledy

Při vytváření aplikací na platformě Android jsou všechny prvky uživatelského rozhraní založeny na využívání pohledů (**View**) a skupin pohledů (**ViewGroup**). Pohled představuje objekt, který se zobrazí na obrazovce a zároveň umožňuje interakci s uživatelem. Skupina pohledů je objekt, který drží jednotlivé pohledy v definovaném pořadí. Toto pořadí je definováno layoutem. (Lacko, 2015)

Uživatelské rozhraní pro každou komponentu vytvářené aplikace je definováno jako hierarchie pohledů a skupin pohledů, jak lze vidět na obrázku 2-8. Každá skupina pohledů pracuje jako neviditelný kontejner, který organizuje své potomky. Jednotlivé pohledy představují prvky uživatelského rozhraní, kterými mohou být text, obrázky, tlačítka atd. (Android Developers, c2017f)



Obrázek 2-8 Ukázka hierarchie pohledů, která definuje layout uživatelského rozhraní, zdroj: Android Developers (c2017f)

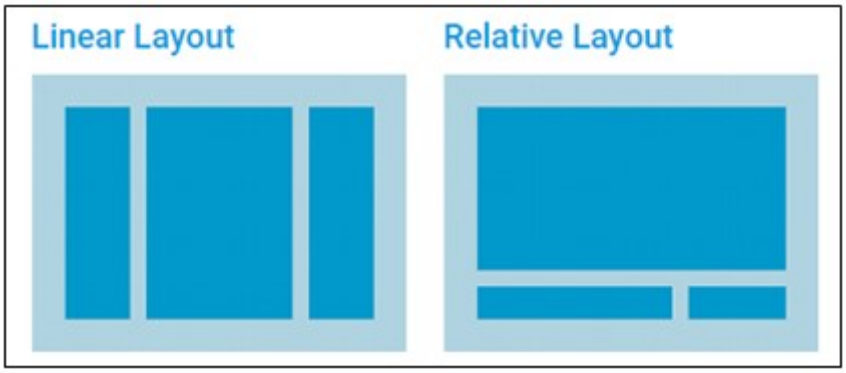
Layout

Layout definuje vizuální strukturu pro uživatelské rozhraní aplikace. Například se může jednat o uživatelské rozhraní pro aktivity. Layout přímo souvisí se skupinou pohledů (ViewGroup), pro které definuje strukturu, v jaké jsou jednotlivé pohledy (View) zobrazeny. Layout je tedy podtřídou třídy ViewGroup. Deklarování layoutu lze provést dvěma způsoby:

- deklarací uživatelských prvků přímo v XML,
- dynamickým vytvářením instancí prvků layoutů za běhu aplikace. (Android Developers, c2017g)

Android framework je flexibilní a umožňuje využití kombinace obou těchto způsobu pro deklaraci a řízení uživatelského rozhraní. Lze tedy deklarovat aplikaci s prvky, které jsou defaultně nastaveny v XML včetně prvků, které se dynamický vytvoří a zobrazí ve chvíli, kdy jsou vyžadovány. (Lacko, 2015)

Každá podtřída třídy ViewGroup umožňuje jedinečný způsob, jak zobrazit jednotlivé pohledy, které se ve skupině nacházejí. Na obrázku 2-9 jsou uvedeny nejběžnější typy layoutů, které se na platformě Android využívají. (Android Developers, c2017g)

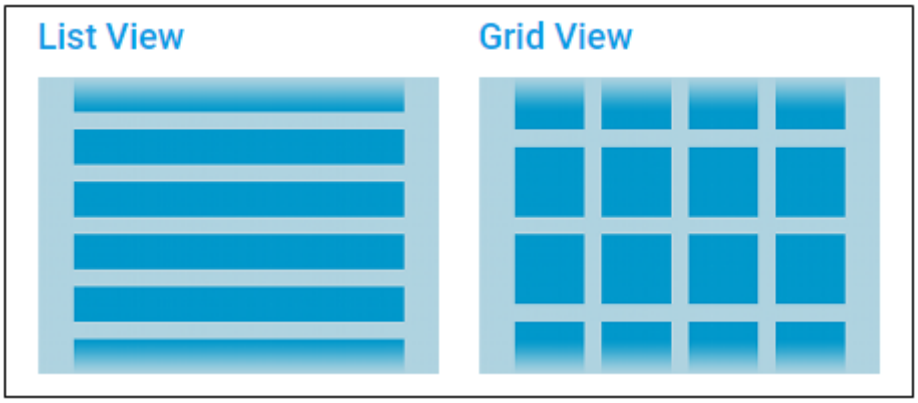


Obrázek 2-9 Ukázka lineárního a relativního layoutu, zdroj: Android Developers (c2017g)

Lineární layout organizuje své potomky do jednoho směru, a to buď vertikálně, nebo horizontálně.

Relativní layout povoluje specifikovat umístění svých potomků relativně ke každému z nich. Například potomek A se nachází nalevo od potomka B. Dále povoluje specifikovat relativní pozici jednotlivých potomků vzhledem k jejich rodičovskému layoutu. Jako příklad lze uvést umístění potomka A na horní okraj. (Android Developers, c2017g)

Pokud je obsah pro daný layout vytvářen dynamicky, lze využít třídu AdapterView, která umožní vytvářet požadovaný obsah za běhu programu. Tato třída využívá adaptér, který prováže požadovaná data s layoutem. Adaptér pracuje jako prostředník mezi zdrojem dat a layoutem, který využívá třídu AdapterView. Prostřednictvím adaptéru jsou načítána data ze zdroje (například kolekce) a následně jednotlivě konvertována do pohledů, které mohou být přidány do layoutu, který využívá třídu AdapterView. Běžné layouty využívající adaptér jsou uvedeny na obrázku 2-10. (Android Developers, c2017f)



Obrázek 2-10 Ukázka ListView a GridView, Android Developers (c2017g)

ListView je skupina pohledů (ViewGroup), která zobrazuje seznam s jednotlivými položkami. Jedna položka je reprezentována konkrétním pohledem (View).

GridView je skupina pohledů, která zobrazuje položky v dvojrozměrné mřížce.

Fragmenty

Fragment reprezentuje chování uživatelského rozhraní pro aktivity. Lze využít kombinaci několika fragmentů pro jednu aktivitu k vytvoření více částí uživatelského rozhraní. Jednotlivé fragmenty je pak možné v rámci aplikace znovu využívat napříč různými aktivitami. Fragment si lze představit jako modulární část aktivity, která má vlastní životní cyklus, přijímá své vlastní vstupní události a lze ji přidat či odebrat kdykoliv za běhu aktivity. Důležité je poznamenat, že fragmenty musí být komponovány aktivitou a jejich životní cyklus je přímo ovlivněn životním cyklem hostující aktivity. (Android Developers, c2017h)

2.3.4 Verze Androidu a jejich tržní podíl

V tabulce 3-1, převzaté z Android Developers (2017i), jsou uvedeny aktuální statistiky tržních podílů jednotlivých verzí operačního systému Android. Tento tržní podíl se mění každým dnem, například verze Nougat během uplynulých dvou týdnu dokázala svůj procentní podíl téměř zdvojnásobit.

Tabulka 2-1: Tržní podíl verzí OS Android, zdroj: Android Developers, (c2017i), upraveno

Přehled tržních podílů jednotlivých verzí operačního systému Android		
Verze	**Název**	**Podíl na trhu**
2.3.3-4.0.4	Gingerbread až Ice Cream Sandwich	1,8%
4.1-4.3	Jelly Bean	10,1%
4.4	Kitkat	20,0%
5.0-5.1	Lollipop	32,0%
6.0	Marshmallow	31,2%
7.0	Nougat	4,9%

2.3.5 Vývojové nástroje a prostředí

Vývoj mobilních aplikací na platformě Android, se stává jednodušší, protože stále více společností vytváří vlastní nástroje, které poskytují vývojářům téměř neomezené možnosti. V současné době je nejvyužívanější vývojové prostředí Android Studio od společnosti Google. Zahrnuje celou řadu užitečných nástrojů pro vývoj mobilních aplikací. Jelikož jsou mobilní aplikace na této platformě vyvíjeny v jazyce Java, je nutné využívat JDK (Java Development Kit). Pro vývoj na konkrétní zařízení je nutné stáhnout konkrétní SDK (Software Development Kit), které je pro toto zařízení určeno. (Mullis, 2016)

Java Development Kit (JDK)

JDK je softwarové prostředí určené pro vývoj Java aplikací a appletů. To zahrnuje Java Runtime Enviroment (JRE), které je nutné pro spouštění aplikací a nástrojů. Dále obsahuje kompilátor, debugger, archivační nástroj, generátor dokumentace atd. (Techopedia, c2015)

Software Development Kit (SDK)

Jedná se o sadu nástrojů, které umožňují vytvářet aplikace pro konkrétní operační systémy a hardware. Pro vývoj na platformě Android se používá Android SDK, které zahrnují:

- vyžadované knihovny,
- debugger,
- emulátor,

- relevantní dokumentaci pro konkrétní typ aplikačního programového rozhraní,
- ukázkový zdrojový kód.

Android studio disponuje nástrojem, který umožnuje rychlou možnost stažení požadovaného SDK. Pro jednotlivé verze Androidu se využívají různé SDK. (Android Studio, c2017)

Android Studio

Android Studio je oficiální integrované vývojové prostředí (IDE) pro vývoj mobilních aplikací na platformě Android založené na IntelliJ IDEA. Android Studio nabízí funkce, které zvyšují produktivitu a efektivitu při budování aplikací. Jako příklad těchto funkcí lze uvést:

- flexibilní využívání nástroje Gradle,
- rychlý a funkcionálně rozsáhly emulátor pro testování aplikací na různých zařízeních,
- jednotné prostředí určené pro vývoj aplikací pro všechny zařízení na platformě Android,
- nástroj okamžitého spuštění sloužící k aplikaci změn bez nutnosti vytvářet nový aplikační balíček (APK),
- rozsáhle testovací nástroje a frameworky,
- zabudovaná podpora pro platformu Cloud od společnosti Google. (Android Studio, c2017)

2.3.6 Možnosti distribuce mobilní aplikace

Android jako otevřená platforma nabízí mnoho možností, jak své aplikace distribuovat mezi uživatele. Může se jednat o zveřejňování prostřednictvím aplikace Google Play, která funguje jako „*tržiště*“. Dále lze využít distribuci skrze webové stránky nebo emailovou komunikaci, kdy aplikaci obdrží cílení uživatelé. Vše je závisle na konkrétním účelu aplikace a volbě vývojáře. (Android Developers, c2017j)

Distribuce prostřednictvím aplikace Google Play

Aplikace Google Play představuje primární „*tržiště*“ pro aplikace na platformě Android. Pokud je vytvořená aplikace určena velkému okruhu uživatelů, pak je

vhodné využít tento způsob distribuce. Naneštěstí není tato distribuce zdarma a je založena na licenčních podmínkách. Na druhou stranu tato licenční ochrana zabraňuje neoprávněnému přístupu a využívání aplikace. (Android Developers, c2017j)

Distribuce aplikace prostřednictvím emailu

Jednoduchý a rychlý způsob jak zveřejnit aplikaci, je poslat ji uživatelům prostřednictvím emailu. Před tímto krokem je nutné aplikaci připravit pro zveřejnění a připojit k příloze e-mailové zprávy. Jakmile uživatel otevře tuto zprávu na zařízení, které je rozpoznáno operačním systémem Android, tak se uživateli zpřístupní soubor typu APK a následně je vyzván k instalaci. (Android Developers, c2017j)

2.4 Technologie a nástroje využité při vývoji aplikace

V této kapitole jsou stručně popsány základní technologie, které byly využity při vývoji mobilní aplikace. Jedná se o objektově orientovaný programovací jazyk Java a rozšiřitelný značkovací jazyk XML (eXtensible Markup Language).

2.4.1 Java

Java je vysokoúrovňový programovací jazyk vyvinut společností Sun Microsystems, inc. a v současné době je spravován společností Oracle Corporation. Jedná se o objektově orientovaný programovací jazyk, který se momentálně nachází ve verzi 8. Tento programovací jazyk se vyznačuje svou univerzálnosti, jelikož není přesně určen pro jednu aplikační oblast. Jinými slovy lze prostřednictvím tohoto programovacího jazyka vytvářet celou řadu aplikaci, ať už se jedná o desktopové, mobilní či webové.

Program napsaný v Javě je přenositelný mezi jednotlivými platformami na úrovni zdrojového i zkompilovaného kódu. Programy jsou nejdříve kompilovány do „*bytekódu*“, a ten je poté kompilován do strojového kódu prostřednictvím JIT kompilátoru. Běh programu je zajištěn prostřednictvím virtuálního stroje známým pod názvem JVM (Java Virtual Machine). Vzhledem k tomu, že JVM je k dispozici na mnoha platformách, stačí vytvořit aplikaci pouze jednou a následně lze spustit na jakékoliv platformě, pro kterou je JVM vyvinuto. Tím je dosaženo obrovské portability, která je klíčovou vlastnosti programovacího jazyku Java.

2.4.2 XML

Extensible Markup Language lze přeložit jako rozšířitelný značkovací jazyk. Jedná se o sadu pravidel pro kódování dokumentů ve strojově čitelné formě. XML je populární formát pro sdílení dat na internetu. Důležité části dokumentu se označují pomocí značek, které se nazývají elementy. Jednotlivé elementy lze do sebe vnořovat a tak zachytit potřebnou strukturu dokumentu.

Při využívání XML na platformě Android lze jednoduše a rychle navrhnout uživatelské rozhraní a potřebný layout. Postup je analogický k používání XML elementu při vytváření webových stránek.

Jazyk XML neobsahuje předem definované značky. Proto je nutné definovat vlastní značky, které se budou využívat. Ty lze definovat v souboru DTD (Document Type Definition). Potom je možné automaticky kontrolovat, zda je vytvářený XML soubor v souladu s definovaným souborem a ověřit tak validitu dokumentu. DTD je definiční jazyk pro strukturu dokumentu XML.

3 Analýza současného stavu podniku v oblasti komunikace

Cílem této kapitoly je představit společnost, pro kterou je tato diplomová práce realizována. Dále seznámit s technologickými možnostmi komunikace, kterými průmyslové pračky v současné době disponují. Nakonec stručně popsat stávající způsob monitorování průmyslových praček a sběru požadovaných dat.

3.1 Popis firmy

Alliance Laundry CE je jednou z dceřiných společností Alliance Laundry Holdings LLC. V současné době se jedná o lídra ve vývoji, výrobě a prodeji prádelenské techniky na celém světě. Tyto produkty jsou hojně využívány v samoobslužných prádelnách, komerčních prádelnách, v hotelech a na řadě dalších míst. Hlavním produktem této společnosti jsou tedy průmyslové pračky. Mezi další důležité produkty lze zmínit například žehliče nebo sušičky.

Společnost Alliance Laundry, pro niž je tato diplomová práce realizována má sídlo ve městě Příbor v Moravskoslezském kraji, kde pracuje více než tisíc zaměstnanců. Kromě běžných oddělení, kde probíhá každodenní výroba má tato firma i vlastní vývojové oddělení, které je hlavní páteří celé společnosti a se svými neustálými inovacemi zajišťuje společnosti pozici jedničky na světovém trhu v oblasti prádelenské techniky.

Vývojové oddělení je rozděleno na dvě základní části. První část tvoří konstrukční oddělení, které se zabývá zajištěním technické podpory pro výrobu praček a to zejména návrhem projektové a konstrukční části. Oddělení, kde je tato práce realizována, se zabývá vývojem softwaru do jednotlivých zařízení. Pro účely této práce jsou důležité právě průmyslové pračky a vývoj softwaru a technologií, které využívají. Na obrázku 3-1 lze vidět fotografii z vývojové haly.

Obrázek 3-1 Vývojová hala, zdroj: vlastní

3.2 Komunikace a technologie průmyslových praček

V tuto chvíli lze konstatovat, že průmyslové pračky vyvíjené v této firmě jsou na vysoké úrovní, co se běžné využitelnosti týče. Jednotlivé programy jsou vysoce optimalizované, aby budoucím zákazníkům v první řadě snížili náklady a v druhé řadě také co nejvíce usnadnili obsluhu těchto zařízení.

Nicméně s nástupem mobilních technologií je stále více a více kladen důraz na schopnost komunikovat na dálku prostřednictvím například mobilních telefonů, kterou by v současné době měly moderní zařízení splňovat. Právě v oblasti komunikace společnost lehce zaostává oproti konkurenci, a proto zde aktuálně probíhá mohutný vývoj. Momentálně jediný způsob, kterým lze s průmyslovou pračkou komunikovat na dálku je prostřednictvím technologie Bluetooth.

Jednotlivé pračky obsahují komunikační moduly včetně Bluetooth senzorů, které využívají technologií BLE (viz kapitola 2.1.4) jako prostředek komunikace. Na základě výše uvedeného, lze považovat správně fungující komunikační modul, za předpoklad pro tuto práci.

3.3 Monitorování a sběr dat

Oblast monitorování a sběru dat je v tuto chvíli řešena velmi jednoduše. Na vývojovém oddělení pracují servisní pracovníci, jejichž primárním úkolem je sledovat jednotlivá zařízení při chodu a ručně pak zapisovat jednotlivé údaje, které například jednotlivé prací cykly vygenerují.

Jelikož se na vývojové hale nachází spousta zařízení, je tento proces poněkud neefektivní a zdlouhavý. Servisní mobilní aplikace, která je náplní této práce by měla toto úskalí jednoznačně zefektivnit a poskytnout tak samotným vývojářům rychlou zpětnou vazbu při tvorbě nových a nových programů.

Testování vyvíjené aplikace

Při vývoji mobilní aplikace je nutné pravidelné testování nově implementovaných funkcí. Pro tyto účely slouží testovací panel průmyslové pračky, který prostřednictvím vloženého softwaru simuluje běžný chod praček. Na obrázku 3-2 je uveden testovací panel, ve kterém je mimo jiné integrován Bluetooth komunikační modul. Pro úpravu softwaru v testovacím panelu slouží USB připojení, prostřednictvím kterého je do panelu nahrán požadovaný software. Tento interní software není blíže popsán v této práci, jelikož se jedná o soukromé know-how společnosti. Testovací panel obsahuje obrazovku a klávesnici pro podrobnější ovládání.

Obrázek 3-2 Testovací panel, zdroj: vlastní

4 Návrh a implementace vyvíjené aplikace

Cílem této kapitoly je popsat celý vývojový proces servisní mobilní aplikace. Vývoj je založen na metodice UP (kapitola 2.2.1), která je pro účely projektu lehce modifikována a jsou využity jen ty části, které mají pro vývoj požadované aplikace smysl. Pří vývoji jsou využívány také prvky agilního vývoje (kapitola 2.2.3) včetně sady technik UX (kapitola 2.2.4).

Tato kapitola je rozdělena na čtyři hlavní podkapitoly, které představují jednotlivé iterace vývoje. Na začátku každé iterace jsou definovány požadavky, které jsou pro její dokončení nezbytné. Tyto požadavky jsou poté analyzovány a zpracovány do konkrétní podoby. Jakmile jsou jednotlivé požadavky detailně strukturovány, je provedena konstrukční část vývoje v podobě převedení požadavků do programového kódu. Na konci každé iterace probíhá validace a předání, kdy je výstup předveden zadavateli ke schválení a případným připomínkám.

4.1 Sběr informací pro návrh aplikace

První iterace představuje sběr důležitých informací, které jsou nezbytné ke správnému návrhu aplikace. Jedná se o samotný účel aplikace, user stories, zpracování uživatelských požadavků a popis základních procesů. Jedním z výstupů této iterace je hrubý přehled požadavků, které by měla vyvíjena aplikace splňovat včetně stručné analýzy. Dalšími výstupy jsou UML diagramy, které slouží mimo jiné pro lepší pochopení jednotlivých funkcionalit mobilní aplikace.

4.1.1 Účel aplikace

Servisní mobilní aplikace slouží k monitorování průmyslových praček prostřednictvím technologie BLE. Hlavním účelem je rychlý zisk důležitých informací o zařízení, jednotlivých programech, cyklech a dalších informací. Aplikace by tedy měla zajistit výraznou úsporu času pro jednotlivé pracovníky a tím i zvýšit jejich celkovou efektivitu.

4.1.2 Uživatelský příběh

Na základě spolupráce s jedním servisním zaměstnancem je zpracován krátký uživatelský příběh. Tento zaměstnanec pracuje na vývojové hale a jeho hlavní pracovní náplní je monitorování a zapisování informací o testovaném softwaru.

Cílem této formy sběru požadavků je, aby uživatel nezávisle popsal, co by měla daná aplikace dělat a jak měla vypadat. Tématem tohoto příběhu je využití mobilní aplikace při monitorování praček. Stručně ořezaná odpověď vypadá následovně:

„*Pokud bych měl k práci využívat mobilní telefon a nějakou aplikaci, rozhodně bych uvítal, aby byla v češtině. Důležité pro mě je, aby byla jednoduše ovladatelná a přehledná.*“

Z tohoto krátkého odstavce lze jasně vydedukovat následující požadavky:

- aplikace by měla být k dispozici v češtině,
- ovládání by mělo být intuitivní,
- jednoduchý a přehledný grafický design.

4.1.3 Seznam požadavků od zadavatele

Na základě několika úvodních konzultací byl vytvořen hrubý přehled požadavků, které by měla mobilní aplikace disponovat. Tyto požadavky jsou následovné:

- navázat spojení s průmyslovou pračkou,
- zobrazení základních informací o pračce,
- zobrazit informace o jednotlivých cyklech,
- zobrazení auditních informací o pračce.

Lze si všimnout, že zadavatel a uživatel se v popisu požadavků jednoznačně liší. Proto je v praxi užitečné používat výše uvedené uživatelské příběhy, jejímž prostřednictvím lze získat další možný úhel pohledu na finální produkt.

4.1.4 Analýza získaných požadavků

Ze získaných požadavků, které byly po několika konzultacích ujasněny, je vytvořen přehled v podobě tabulky se stručnou analýzou ke každému požadavku. V tabulce 4-1 je možné tento přehled vidět. Tato tabulka slouží i jako první výstup pro zadavatele a je jistou specifikací zadaných požadavků.

Tabulka 4-1 Hrubý přehled požadavků, zdroj: vlastní

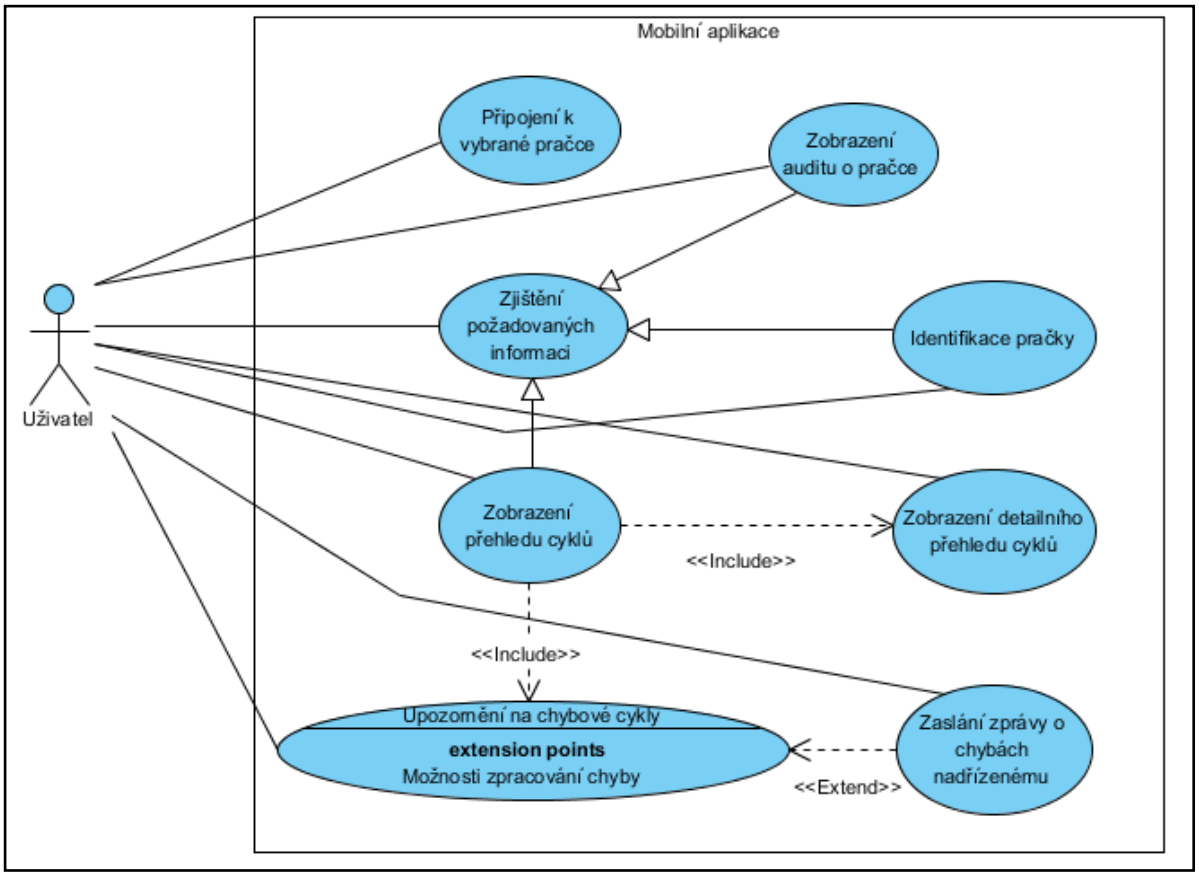
Tabulka požadavků	
Požadavek	**Analýza**
Navázaní spojení s průmyslovou pračkou	Tento první požadavek by již měl být z jisté části rozpracován, jelikož komunikační modul obsahuje i vzorovou aplikaci, kde je toto spojení v jisté formě implementováno. Pro komunikaci je využita technologie BLE.
Zobrazení základních informací o konkrétní pračce	Jedná se o jednu obrazovku, která zobrazí základní informace. Tuto obrazovku lze reprezentovat aktivitou nebo fragmentem (kapitola 2.3.3).
Zobrazení informací o jednotlivých cyklech.	Jelikož na jednotlivých pračkách probíhá velký počet cyklů, je nutno zvolit dynamický způsob vytváření obsahu obrazovky. Pro účely této obrazovky je nutné využít layout s adaptérem, konktrétně ListView (kapitola 2.3.3).
Zobrazení auditních informací o připojené pračce.	Auditní informace představují například celkový počet cyklů, které na pračce proběhly apod. Postup pro tuto obrazovku je analogický k analýze druhého požadavku.
Překlad aplikace do češtiny.	Podpora pro multijazyčnost je zajištěna s využitím unikátních textových řetězců, které se poté konkrétně lokalizují pro vybranou zemi.
Intuitivní ovládání a jednoduchý grafický design.	Aplikace by měla být co možná nejjednodušší na pochopení a graficky odpovídat oblasti, pro kterou je určena.

4.1.5 Popis mobilní aplikace z pohledu uživatele

Pro základní popis aplikace z pohledu uživatele lze využit diagram případů užití. Prostřednictvím tohoto diagramu lze jednoduše znázornit, jakými funkcemi by měla vyvíjená mobilní aplikace disponovat z pohledu uživatele (zadavatele). Jedná se o diagram jazyka UML, který je popsán v kapitole 2.2.2

Na obrázku 4-1 lze vidět diagram případů užití, který popisuje základní funkcionality vyvíjené mobilní aplikace. Lze si všimnout, že jako aktér vystupuje pouze uživatel. Je to způsobeno tím, že mobilní aplikace ve své podstatě nerozděluje uživatele na role a neposkytuje každé z rolí jiné funkcionality. Samotný systém

na diagramu představuje mobilní aplikace, která uživateli umožnuje funkce na obrázku znázorněny jako případy užití.



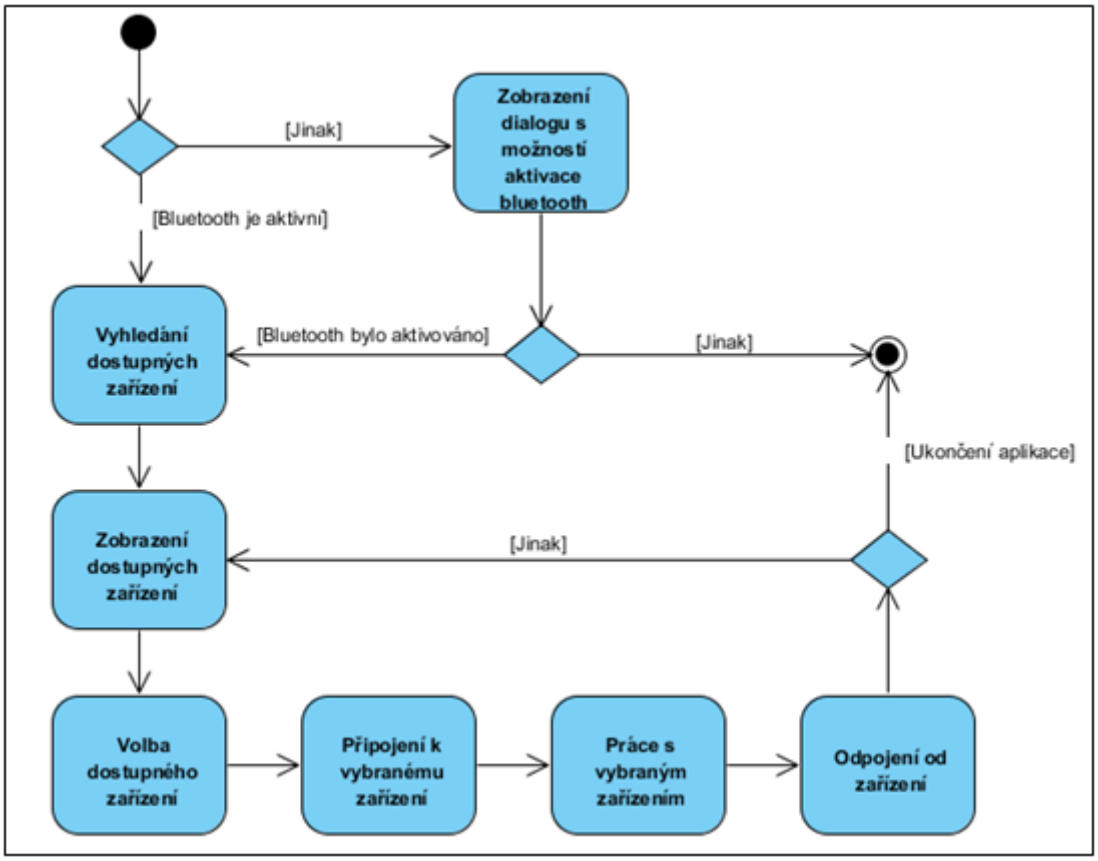
Obrázek 4-1 Diagram případů užití mobilní aplikace, zdroj: vlastní

Z obrázku jsou patrné základní funkcionality, které by měla mobilní aplikace splňovat z pohledu uživatele. Jako hlavní případ užití se jeví „*Zjištění požadovaných informací*“. Z tohoto případu užití se pak odvíjí funkcionality. Je to logické, jelikož primární úkol mobilní aplikace je získat informace z pračky a následně je přehledně interpretovat prostřednictvím uživatelského rozhraní.

4.1.6 Dynamický popis mobilní aplikace

K dynamickému popisu aplikace je využit diagram aktivit. Prostřednictvím tohoto diagramu lze zachytit workflow mobilní aplikace.

Na obrázku 4-2 lze vidět diagram aktivit pro vyvíjenou aplikaci. Důležité je poznamenat, že tento diagram zachycuje základní tok aktivit již pro celou aplikaci, a ne pouze pro první iteraci. To, v čem se budou jednotlivé iterace lišit, je aktivita označena jako „*Práce s vybraným zařízením*“.



Obrázek 4-2 Diagram aktivit mobilní aplikace, zdroj: vlastní

Co se týče samotného diagramu na obrázku 4-2, lze jej popsat dle následujících kroků:

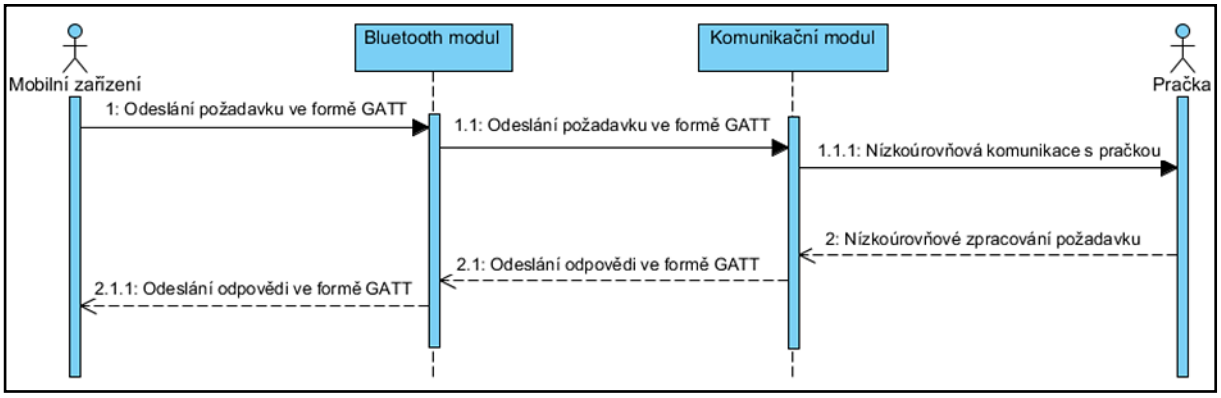
1. **Je Bluetooth k dispozici** - při spuštění mobilní aplikace se jako první provede dotázání, zda je k dispozici Bluetooth. Pokud je Bluetooth aktivní, aplikace se korektně spustí. Pokud je Bluetooth vypnuto, je uživateli nabídnuta možnost jej aktivovat (**Zobrazení dialogu s možností aktivace Bluetooth**). Ve všech jiných případech není možné s aplikací nadále pracovat.
2. **Vyhledávání dostupných zařízení** – v této fázi probíhá vyhledávání zařízení, které jsou k dispozici pro komunikaci.
3. **Zobrazení dostupných zařízení** – následně se zobrazí zařízení, s kterými lze komunikovat.
4. **Volba dostupného zařízení** – uživatel si může zvolit, ke kterému dostupnému zařízení se chce připojit.
5. **Připojení k vybranému zařízení** - poté probíhá samotné připojení k vybranému zařízení a **práce s vybraným zařízením**.

6. **Odpojení od zařízení** - nakonec má uživatel možnost se od zařízení odpojit a například začít pracovat s jiným dostupným zařízením, nebo celou aplikaci ukončit.

4.1.7 Základní popis interakce mezi objekty

Pro základní popis interakce mezi klíčovými objekty, které se v aplikaci vyskytují lze využít sekvenční diagram.

Na obrázku 4-3 lze vidět sekvenční diagram, který popisuje základní způsob, jakým mobilní zařízení komunikuje s pračkou. Z diagramu je patrné, že existují dva klíčové objekty a to „*Bluetooth modul*“ a „*Komunikační modul*“. Díky těmto objektům lze navázat spojení mezi mobilním zařízením a pračkou. Tyto zařízení jsou v diagramu označeny jako aktéři. Samotná technologie pro komunikaci s průmyslovou pračkou se nazývá BLE. Detailní popis lze nalézt v kapitole 2.1.4.



Obrázek 4-3 Sekvenční diagram, zdroj: vlastní

Na obrázku 4-3 lze tedy vidět jednoduché znázornění komunikace mezi mobilním zařízením a pračkou. Průběh komunikace může být následující:

1. **Mobilní zařízení** odešle požadavek ve formě GATT, který specifikuje přesné akce, které chce učinit. V tomto případě se jedná pouze jen o zjištění specifických údajů. Ovšem je také možné například na dálku ovládat zamykání dveří u pračky, což samozřejmě také vyžaduje detailnější zpracování požadavku.
2. **Bluetooth modul** je prostředníkem komunikace mezi mobilním zařízením a pračkou. Prostřednictvím tohoto modulu, který je schopný interakce s komunikačním modulem pračky lze tak navázat komunikaci s pračkou.

3. **Komunikační modul** je součásti pračky. Požadavek, který je směrován na pračku převezme právě tento komunikační modul, prostřednictvím kterého je možné zpracovat požadavek na úrovni **pračky**. Prostřednictvím komunikačního modulu probíhá tedy zpracování požadavku a vytváření odpovědi, která je pak stejnou trasou směrována k mobilnímu zařízení.

Komunikační programový modul obsahuje interní software firmy, pro kterou je aplikace vyvíjena, a tedy nelze jej v této práci uvádět. V další části je tedy předpokládáno, že tento modul je již vytvořen nezávisle na této práci.

4.1.8 Validace

Po získání jednotlivých požadavků a jejich následné analýze je nutno zadavatele obeznámit se zjištěnými údaji. Dále lze také využít vytvořené UML diagramy pro vysvětlení základních funkcionalit mobilní aplikace.

Cílem této první validace je definovat jednotný směr vývoje. Jelikož zadavatel nemá zkušenosti s vývojem mobilních aplikací, je nutné ho obeznámit se základními principy a jednotlivé postupy vysvětlit na obecné rovině.

Jednotlivé analýzy byly zadavatelem schváleny a velmi kladně hodnoceny. Svou roli v tom rozhodně sehrály i UML diagramy, jelikož se tato grafická anotace využívá i při vývoji softwaru pro průmyslové pračky.

4.2 Vytvoření prvního prototypu mobilní aplikace

Jelikož vývoj probíhá agilně, je nutno zadavateli poskytovat pravidelně nějaký výstup a flexibilně tak reagovat na jeho požadavky. V rámci této iterace je vytvořen první velmi zjednodušený prototyp aplikace, který slouží jako návrh celé architektury. Je opět podstatné komunikovat se zadavatelem a přesně specifikovat, jaké funkce má tento první prototypový model implementovat a jak má vypadat.

4.2.1 Specifikace požadavků na první prototyp aplikace

V tuto chvíli je nutno definovat již konkrétní požadavky, které mají být v rámci této iterace provedeny. Jedná se tedy o podrobnější rozpracování konkrétních požadavků z první iterace.

Prvním základním požadavkem je navázání spojení mobilního zařízení s pračkou prostřednictvím technologie Bluetooth. Správná implementace tohoto

požadavku je pro funkci celé aplikace naprosto nezbytná. Druhým požadavkem je zobrazení základních informaci o pračce, která je momentálně propojena s mobilním telefonem.

Aktuálně jsou tyto dva požadavky klíčové pro další vývoj. Podrobný přehled požadavků na první prototyp aplikace lze vidět v tabulce 4-2.

Tabulka 4-2 Přehled požadavků pro první prototyp mobilní aplikace, zdroj: vlastní

Tabulka požadavků		
Požadavek	**Analýza**	**Výstup**
Navázání komunikace s pračkou prostřednictvím technologie bluetooth (BLE).	Je potřeba do vytvářené aplikace implementovat modul, který zajišťuje bluetooth komunikaci mezi zařízeními. Poté je nutné tento modul upravit, aby byl plně kompatibilní pro obě zařízení.	Vytvářená mobilní aplikace je schopna prostřednictvím bluetooth technologie navázat komunikaci s pračkou.
Zobrazení základních informací o vybrané pračce.	Je nutno na obou stranách zařízení implementovat způsob komunikace a zasílaní dat. Tato se práce se zabývá pouze návrhem mobilní aplikace.	Po navázání komunikace s pračkou obsahuje mobilní aplikace aktivitu, která zobrazuje základní informace o pračce.
Vytvoření požadovaného designu aplikace.	Pro návrh designu aplikace je využito sady technik UX.	Diagramy mobilních obrazovek vytvořené prostřednictvím softwaru Visual Paradigm, který je uzpůsobený k podpoře User Experience.

4.2.2 Návrh mobilní aplikace s využitím UX

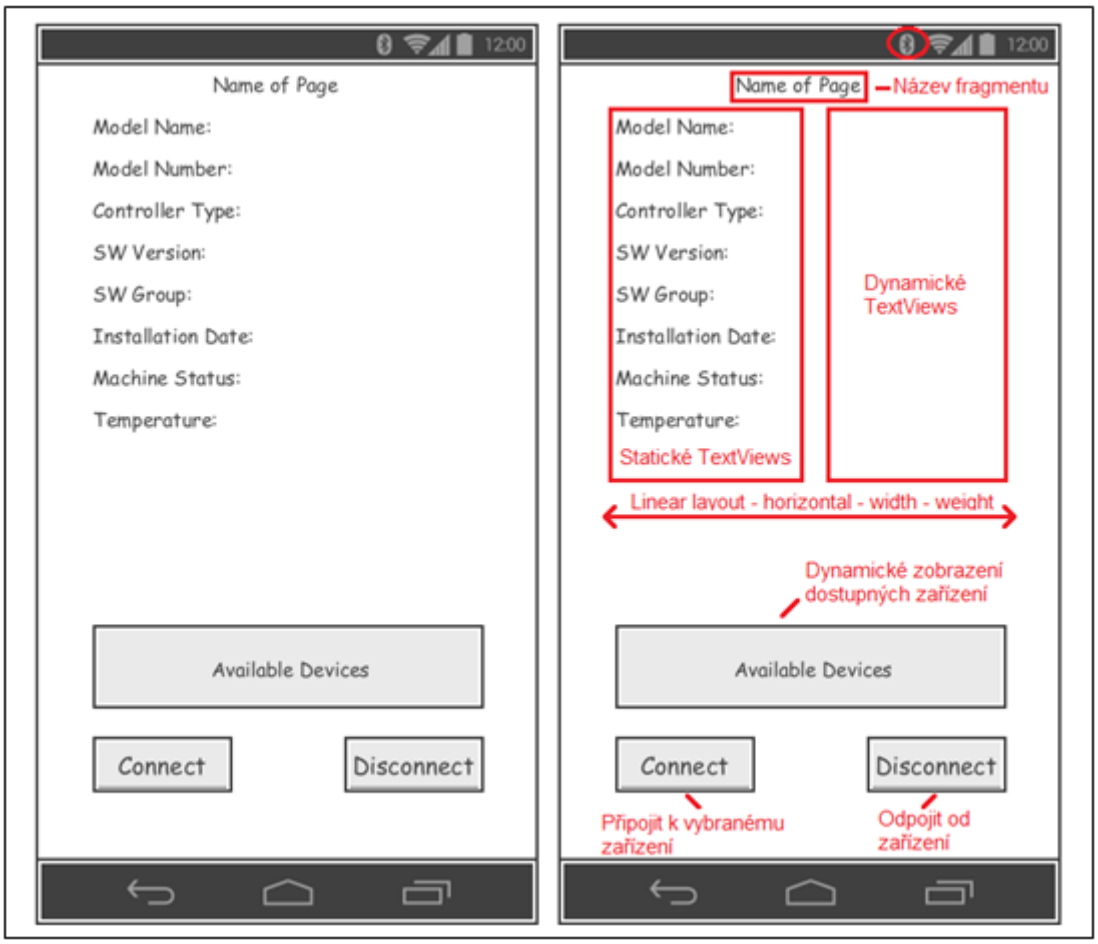
Pro návrh designu mobilní aplikace je využito sady technik UX. Jako podpůrný software pro tuto část je využit program Visual Paradigm, který podporuje návrh mobilních aplikací skrze vytváření tzv. wireframe obrazovek.

Prostřednictvím UX lze agilně komunikovat s uživatelem a aktivně ho tak zapojit do vývojového procesu. Hlavní výhodou je, že uživatel je nápomocný

při návrhu aplikace. To samozřejmě usnadní i konečnou validaci vytvářeného prototypu aplikace.

Návrh úvodní obrazovky mobilní aplikace

Na obrázku 4-4 lze vidět návrh obrazovky samotným uživatelem a také analýzu stejné obrazovky vývojářem. Tato obrazovka uživateli poskytne základní přehled o pračce, ke které je připojen. Charakter informací, které by se měly na obrazovce objevit je zřejmý z obrázku.



Obrázek 4-4 Návrh obrazovky se základními informacemi o pračce včetně analýzy, zdroj: vlastní

Ve spodní části obrazovky se nachází pole, ve kterém se zobrazí dostupná zařízení pro komunikaci při zapnutí aplikace. K vybranému zařízení je potom možné se připojit nebo odpojit prostřednictvím tlačítek. Návrh obrazovky uživatelem je velmi jednoduchý a přehledný, což přesně vystihuje charakter vyvíjené aplikace.

V druhé části obrázku lze vidět obrazovku rozšířenou o jednoduchou analýzu vývojáře této aplikace. V horní části mobilního zařízení je označen symbol Bluetooth,

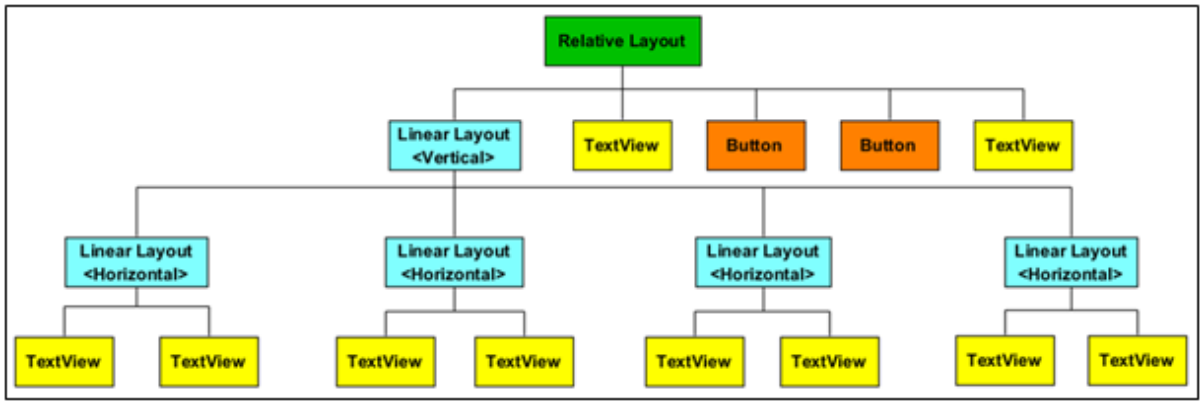
které je nutné pro chod aplikace. Pro zobrazení uživatelem požadovaných údajů o pračce jsou využity TextViews. S tím rozdílem, že tzv. hlavičky pro data jsou vytvářeny staticky v souboru XML. Pozornost je nutno věnovat sekci označené jako „Dynamické Views“. V této části jsou vytvářeny TextViews dynamicky a zobrazují tak údaje, které jsou v průběhu komunikace z pračky získávány. Pro zobrazení a rozmístění těchto statických a dynamických TextViews je využito kombinace lineárního horizontálního a vertikálního layoutu (kapitola 2.3.3).

Co se týče samotného grafického designu aplikace, po několika konzultacích bylo rozhodnuto, že je toto zpracování plně v kompetenci vývojáře aplikace. Musí splňovat jen základní podmínky, kterými jsou jednoduchost, přehlednost a barevně vystihovat prostředí, pro které je aplikace vyvíjena.

4.2.3 Návrh layoutu UI s využitím hierarchie pohledů

V tuto chvíli je již možné pustit se do samotné realizace a konstrukce aplikace. Avšak v praxi se ještě před samotnou tvorbou aplikace využívá diagram hierarchie pohledů, který definuje layout uživatelského rozhraní (kapitola 2.3.3). Tento diagram je vhodné využít při návrhu uživatelského rozhraní a jednotně definovat, jak má layout uživatelského rozhraní vypadat.

Na obrázku 4-5 lze vidět diagram hierarchie pohledů. Jako rodičovský prvek vystupuje relativní layout (ViewGroup).



Obrázek 4-5 Diagram hierarchie pohledů pro obrazovku se základními informacemi, zdroj: vlastní

Pro tabulkové zobrazení charakteristických informací o pračce je využito struktury rodičovského lineárního layoutu (ViewGroup), který dává všem pohledům uvnitř vertikální směr. Tento lineární layout má pod sebou několik horizontálních lineárních layoutů (ViewGroup), které tvoří jednotlivé řádky pomyslné tabulky.

Z důvodu přehlednosti obrázku několik lineárních horizontálních layoutů chybí, nicméně podstata z obrázku jasně vyplývá. Poté jsou zde vidět čtyři pohledy (View), které jsou přímo potomky relativního layoutu, a to z toho důvodu, aby mohly být umístěny do spodní části obrazovky.

4.2.4 Vytvoření prvního fragmentu mobilní aplikace

Pro vytváření prvního fragmentu jsou využity všechny nasbírané informace a analýzy, které z velké části usnadňují konstrukci aplikace.

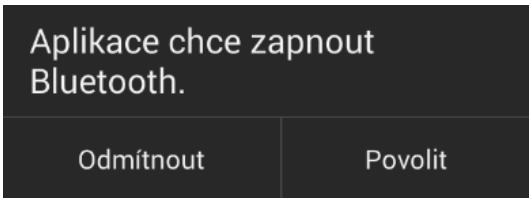
Pro reprezentaci jednotlivých obrazovek se využívají aktivity. Aby bylo však možné dosáhnout lepší interakce mezi jednotlivými obrazovkami a tím výrazně zvýšit ovladatelnost aplikace, je třeba využit fragmentů, které reprezentují chování jednotlivých aktivit (více o aktivitách a fragmentech v kapitole 2.3.3).

Mobilní aplikace obsahuje jednu hlavní aktivitu, která implementuje modul pro Bluetooth komunikaci. V rámci této aktivity jsou komponovány jednotlivé fragmenty. Základní funkcionalita (komunikace s pračkou) mobilní aplikace je tedy postavena na hlavní aktivitě, která pracuje na pozadí. Fragmenty reprezentují jednotlivé obrazovky, které mobilní aplikace obsahuje. Firma, pro kterou je tato práce realizována si nepřeje sdělovat informace o konfiguraci Bluetooth komunikačního modulu. Z toho důvodu není v této práci více přiblížena funkcionalita hlavní aktivity.

Získávání dat z pračky se řídí interním datagramem firmy. Základní princip spočívá v přenosu dat z pračky do mobilního zařízení a jejich následné interpretaci pomocí datagramu do čitelné podoby (informací). Tato interpretace je prováděna na úrovni jednotlivých fragmentů. Například jednotlivé stavy pračky jsou reprezentovány číselnými hodnotami. Tyto číselné hodnoty je potom nutné interpretovat do textové podoby prostřednictvím datagramu. Datagram je uveden jako příloha 2.

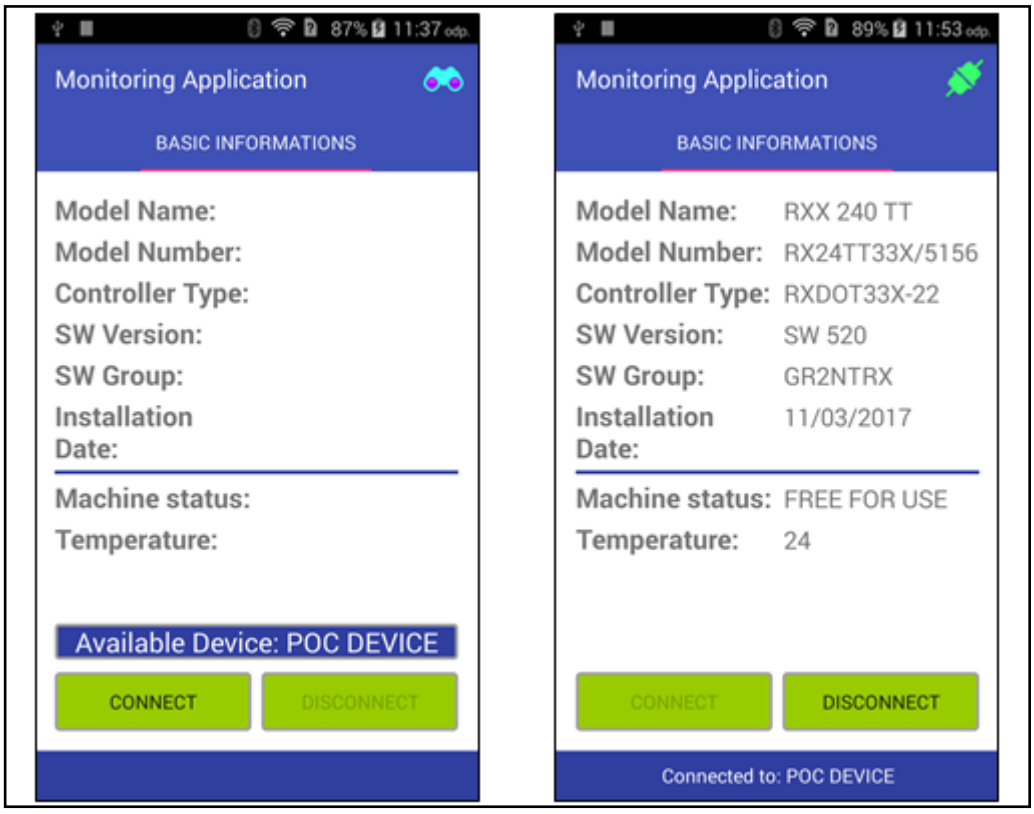
Vytvoření fragmentu pro úvodní obrazovku

Pro správný chod aplikace je potřeba disponovat technologií Bluetooth v mobilním zařízení. V rámci mobilní aplikace jsou nastaveny oprávnění, prostřednictvím kterých je uživatel vyzván k aktivaci Bluetooth. Pokud tak uživatel neučiní, aplikace se nespustí. Detailní dialog lze vidět na obrázku 4-6.



Obrázek 4-6 Bluetooth dialog, zdroj: vlastní

Teď lze již přejít k samotné úvodní obrazovce (fragmentu), která je vidět na obrázku 4-7. Obrázek je rozdělen na dvě části a to před připojením k vybrané pračce a po připojení.



Obrázek 4-7 Úvodní obrazovka se základními informacemi, zdroj: vlastní

V první části obrázku je uvedena aplikace při jejím prvotním spuštění. Nahoře lze vidět název aplikace včetně ikony, která informuje o aktuálním stavu. Ikona připomínající dalekohled symbolizuje vyhledávání dostupných zařízení. Při připojení nebo odpojení se tato ikona dynamicky mění, jak lze vidět například na druhé části obrázku. Nahoře je uveden název fragmentu, který je v tuto chvíli aktivní. Většinu obrazovky pokrývají staticky vytvořené TextViews, jak již vyplývá z návrhu samotného uživatele. Ve spodní části obrazovky se nachází pole, ve kterém se dynamicky zobrazují zařízení, která jsou dostupná pro připojení. Pro připojení a odpojení jsou k dispozici tlačítka v dolní části obrazovky.

Na druhé části obrázku lze vidět první obrazovku (fragment) po připojení. Obrazovka obsahuje dynamicky vytvořené TextViews, které obsahují základní informace o pračce. Na dolním okraji obrazovky se nachází dynamicky upravována patička, která informuje o stavu připojení.

4.2.5 Validace

Po vytvoření úvodní obrazovky (fragmentu) aplikace lze takto vytvořený prototyp předložit zadavateli. Uživatelské rozhraní vychází ze samotného návrhu uživatele a je doplněno o několik nových funkcionalit.

Zadavatel je se samotnou aplikací spokojen a oceňuje především grafický design uživatelského rozhraní a jeho přehlednost. Co se týče funkcionální části, tak jsou splněny všechny požadavky, které byly stanoveny. Do budoucna lze takto vytvořený fragment rozšířit například o nové informace o pračce apod. První prototyp aplikace je tedy úspěšně předán a je možné pokračovat v dalším vývoji.

4.3 Vytvoření druhého prototypu mobilní aplikace

Po vytvoření a předání prvního funkčního prototypu aplikace je třeba začít v uvozovkách zase od začátku. Jinými slovy opět s uživatelem podrobně definovat, kde se má ubírat další vývoj aplikace. Tento další vývoj je postaven již na fungujícím prototypu aplikace, který je výstupem předchozí iterace.

4.3.1 Specifikace požadavků na druhý prototypu mobilní aplikace

Na začátku je nutné znovu stanovit požadavky, které musí výsledný prototyp mobilní aplikace splňovat. Jedná se o rozšíření aplikace o další dvě části. První z těchto části je obrazovka, jejímž obsahem je základní přehled informací o všech testovacích pracích cyklech, které doposud proběhly. Druhou částí je obrazovka, kterou lze nazvat jako celkový audit pračky obsahující informace sumarizačního charakteru. Například celkový počet cyklů, které na pračce proběhly, počet nedokončených cyklů, celkový prací čas v minutách atd. Tyto údaje mají pro uživatele význam při následných analýzách, kdy například zjišťuje procentní chybovost jednotlivých pracích programů.

V tabulce 4-3 lze je uveden podrobný přehled požadavků včetně stručné analýzy. Co se týče samotného návrhu designu aplikace, opět je jako v předchozí iteraci využito sady technik UX.

Tabulka 4-3 Přehled požadavků na druhý prototyp mobilní aplikace, zdroj: vlastní

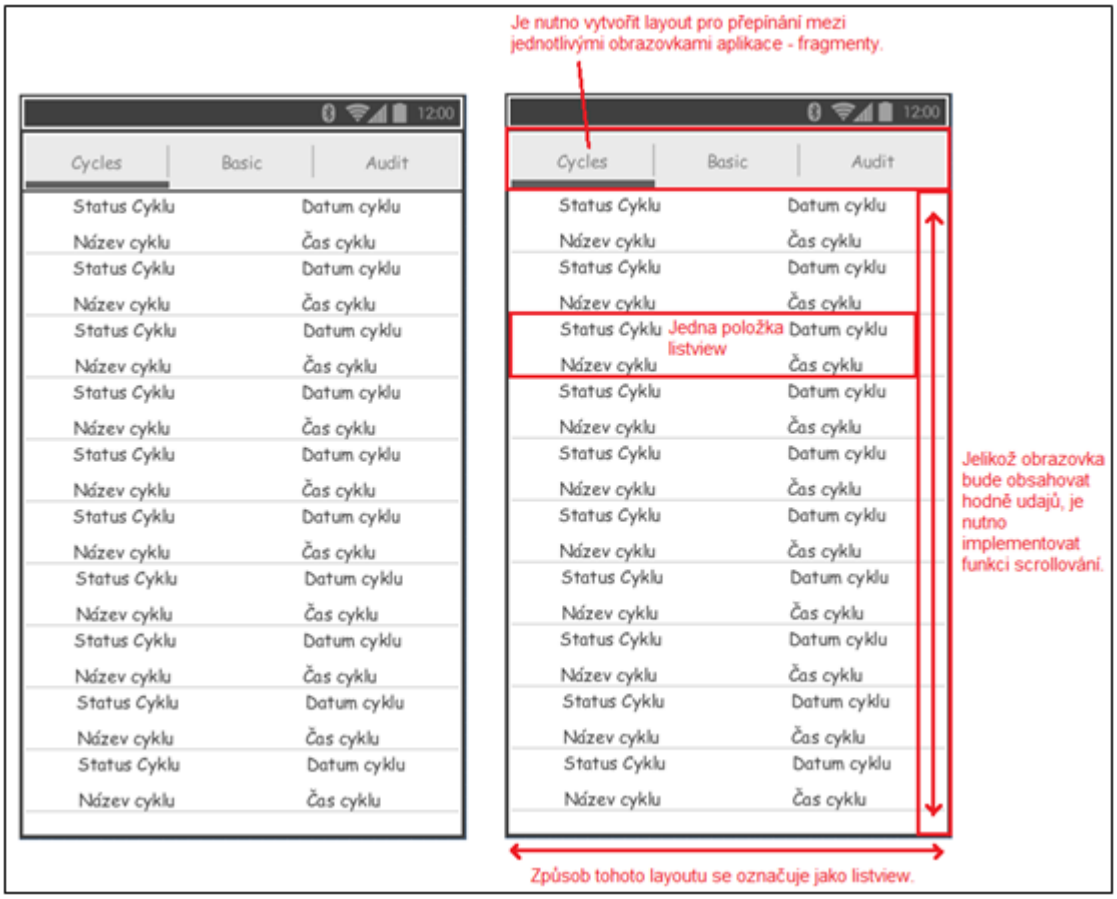
Tabulka požadavků		
Požadavek	**Analýza**	**Výstup**
Vytvoření stručného přehledu všech cyklů, které na dané pračce již proběhly. Každý cyklus musí obsahovat základní charakteristický popis.	Jelikož na pračce může proběhnout mnoho cyklů, je třeba zvolit specifický způsob ukládání získaných dat z pračky a jejich následného zobrazení.	Obrazovka (fragment), který uživateli poskytne přehled o dosavadních cyklech.
Vytvoření obrazovky, která obsahuje auditní informace o konkrétní pračce.	Pračka obsahuje velké množství auditních informací, a proto je nutné implementovat pro obrazovku posunovací mechanismus.	Obrazovka (fragment), která jednoduchým způsobem zobrazí detailní auditní informace.

4.3.2 Návrh druhého prototypu mobilní aplikace s využitím UX

Tato podkapitola je rozdělena na dvě části. První část je věnována návrhu obrazovky sloužící k zachycení přehledu jednotlivých cyklů, které na pračce proběhly. V druhé části je popsán návrh obrazovky s auditními informacemi.

Návrh obrazovky pro zobrazení přehledu cyklů

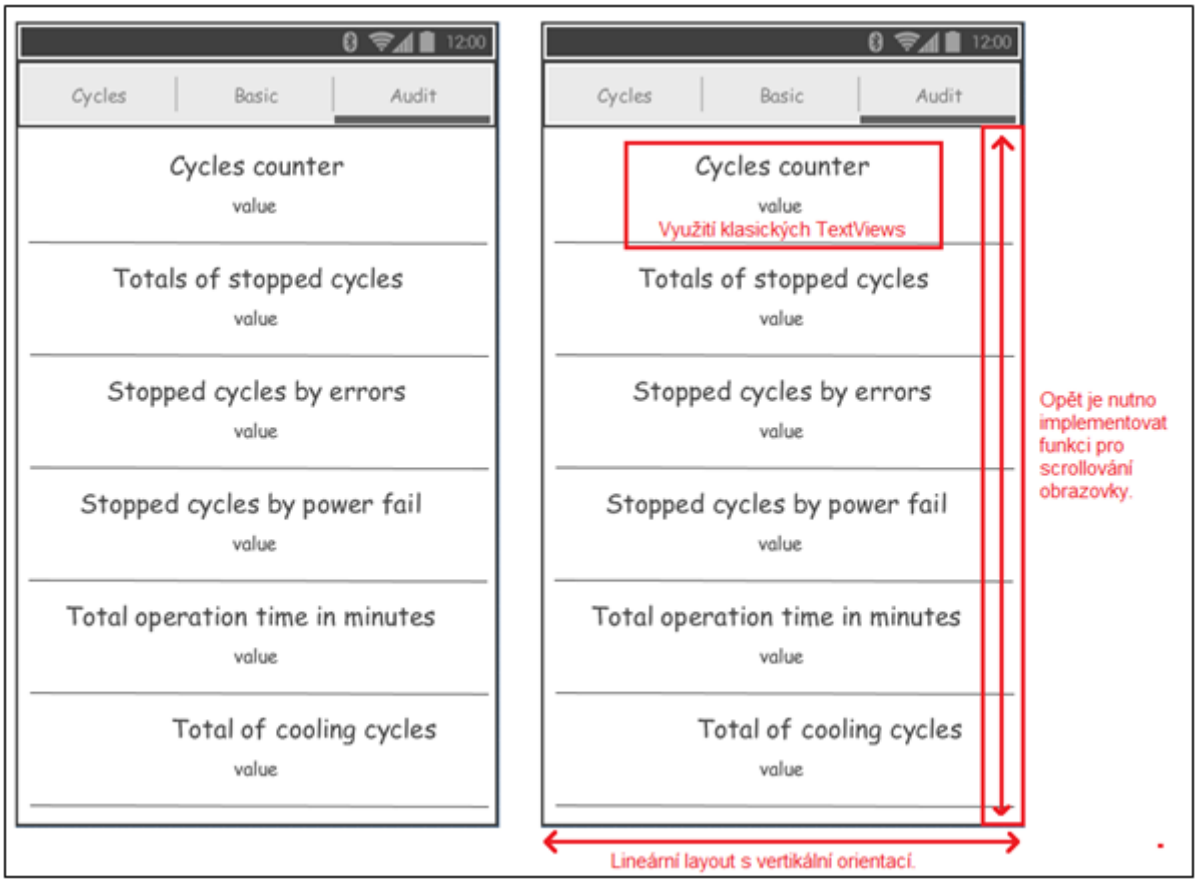
Na obrázku 4-8 je uveden návrh obrazovky uživatelem včetně stručné analýzy vývojáře. Tato obrazovka (fragment) slouží k zobrazení základního přehledu informací o jednotlivých cyklech. Již na první pohled je patrné, že se jedná o stále stejný charakter informací, které akorát mění svoji hodnotu. Nejefektivnější způsob, jak zachytit uživatelské rozhraní této obrazovky je prostřednictvím ListView (kapitola 2.3.3). Jedna položka tohoto layoutu je tedy reprezentována několika TextViews. Jelikož se jedná o přidání nového fragmentu, je třeba implementovat způsob pro přepínání mezi jednotlivými fragmenty. Uvedená obrazovka musí také implementovat způsob pro posouvání (ScrollView). Jednotlivé cykly jsou seřazeny podle data a času.



Obrázek 4-8 Návrh obrazovky pro zobrazení přehledu cyklů, zdroj: vlastní

Návrh obrazovky pro zobrazení auditních informací

Pro návrh obrazovky, prostřednictvím které jsou zachyceny auditní informace o pračce, je využito stejného postupu. Na obrázku 4-9 je uveden návrh od uživatele včetně analýzy vývojáře. Přesto, že je tento návrh velmi jednoduchý, je pro aplikaci naprosto dostačující. Cílem této obrazovky je uživateli poskytnout základní přehled o testované pračce. Ze zjištěných informací lze jednoduše zjistit, jak dlouho se již provádí testování konkrétního cyklu, jeho chybovost apod. Z těchto informací lze vyvodit podrobné analýzy, které uživateli pomůžou při vývoji nového softwaru do praček. Samotný návrh layoutu uživatelského rozhraní je velmi jednoduchý. Jedná se o kombinaci běžného lineárního a vertikálního layoutu.



Obrázek 4-9 Návrh obrazovky pro zobrazení auditních informací, zdroj: vlastní

Auditní informace zachyceny touto obrazovkou mohou být následující:

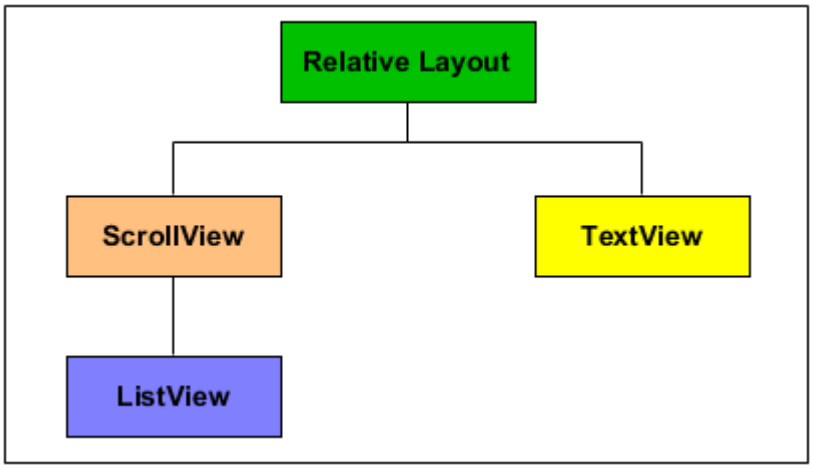
- celkový počet cyklů, které na stroji již proběhly,
- celkový počet nedokončených cyklů,
- nedokončené cykly z důvodu chybového stavu,
- nedokončené cykly z důvodu výpadků elektřiny,
- celkový operační čas pračky uvedený v minutách,
- celkový počet cyklů, které jsou zaměřeny na chlazení.

4.3.3 Návrh layoutu UI s využitím hierarchie pohledů

Před samotnou konstrukcí aplikace je vhodné sestavit diagramy hierarchie pohledů (kapitola 2.3.3). Tento postup je v praxi při vyvíjení mobilních aplikací naprostou nezbytností a samozřejmostí. Podkapitola je rozdělena do dvou částí. Nejdříve je uveden diagram hierarchie pohledů pro obrazovku (fragment) obsahující přehled cyklů a následně poté pro obrazovku s auditními informacemi.

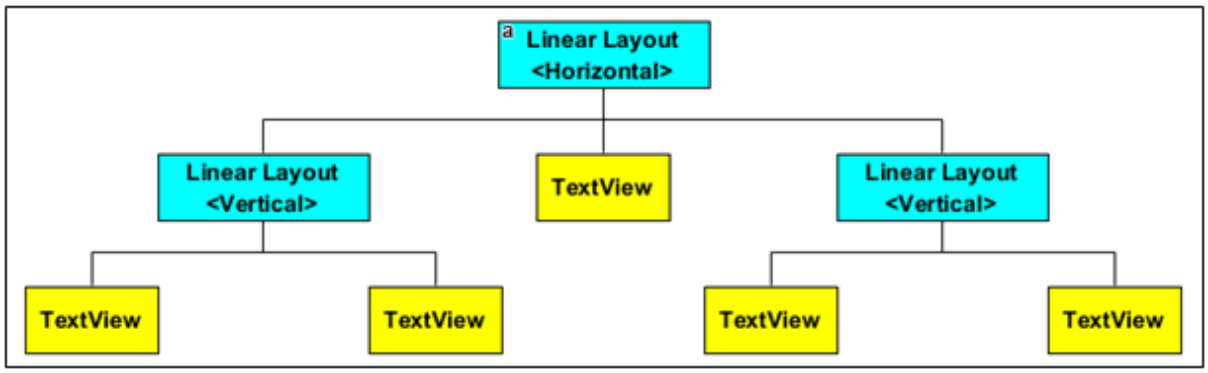
Diagram hierarchie pohledů pro obrazovku s přehledem cyklů

Na obrázku 4-10 je uveden diagram, který zobrazuje hierarchii pohledů layoutu uživatelského rozhraní. Lze si všimnout, že obsahuje dvě nové skupiny pohledů (ViewGroup). Jedná se o ScrollView, umožňující posunovat obrazovku ve vybraném prostoru. Druhou skupinou pohledů se nazývá ListView (viz kapitola 2.3.3).



Obrázek 4-10 Diagram hierarchie pohledů pro obrazovku s přehledem cyklů, zdroj: vlastní

Při využívání ListView je nutné implementovat hierarchii pohledů, která definuje uživatelské rozhraní pro jednu položku této skupiny pohledů. Na obrázku 4-11 lze tedy vidět konkrétní zobrazení jedné položky, kterou ListView obsahuje. Při tomto návrhu je vycházeno ze základního předpokladu, aby jednotlivé položky působily přehledně, a navíc zabíraly pokud možno co nejméně prostoru na obrazovce. Pro zobrazení je využito lineárního layoutu.

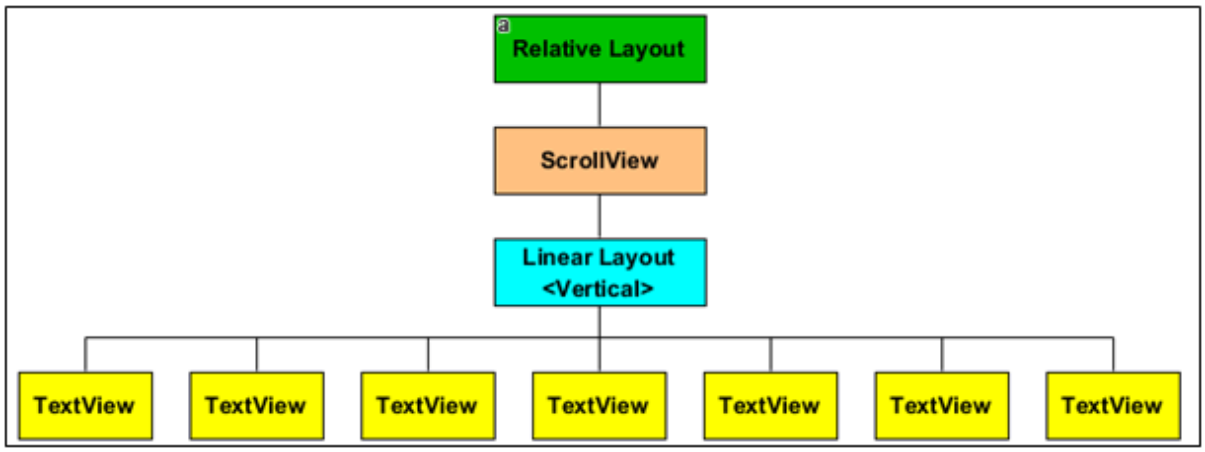


Obrázek 4-11 Diagram hierarchie pohledů pro jednu položku ListView, zdroj: vlastní

Diagram hierarchie pohledů pro obrazovku s auditními informacemi

Diagram hierarchie pohledu pro zachycení layoutu uživatelského rozhraní této obrazovky je vytvořen velmi jednoduše. Obrazovka opět obsahuje ScrollView, jelikož

je potřeba obrazovku posunovat (scrollovat). Hlavní část obrazovky je organizována pomocí lineárního layoutu s vertikální orientací, který obsahuje TextViews pro zobrazení auditních informací, jak je patrné z obrázku 4-12 .



Obrázek 4-12 Diagram hierarchie pohledů pro obrazovku s auditními informacemi, zdroj: vlastní

Schopnost převést uživatelem navržené obrazovky do výše uvedených diagramů je v moderním vývoji mobilních aplikací základní předpoklad k dosažení spokojenosti na obou stranách. Dle těchto diagramů je zkušený programátor bez problémů schopný převést navrženou aplikaci do podoby finálního programového kódu.

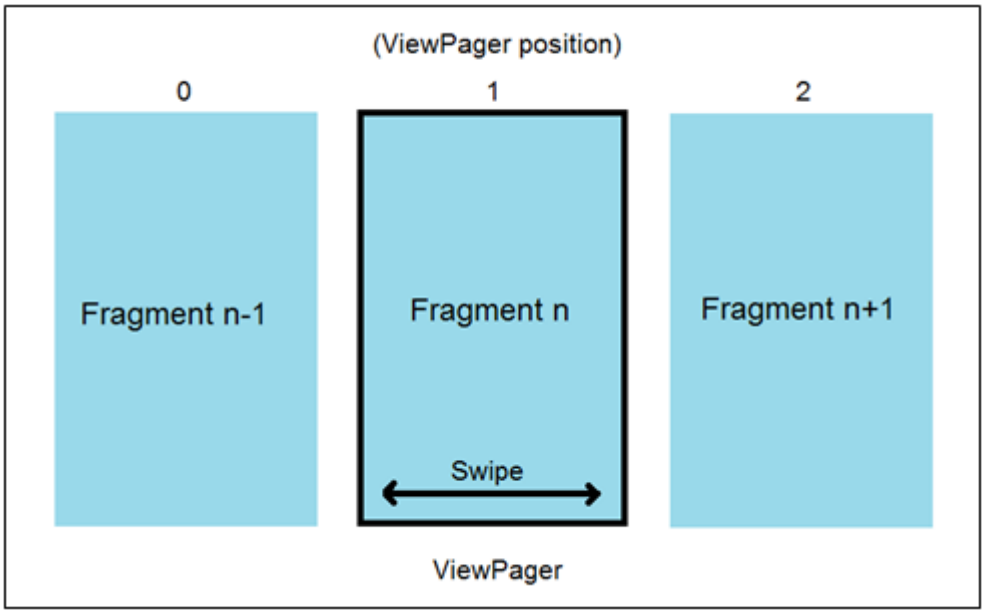
4.3.4 Vytvoření fragmentů pro druhý prototyp mobilní aplikace

V rámci této podkapitoly je popsán postup konstrukce při vytváření fragmentů pro zobrazení přehledu cyklů a auditních informací o pračce. Správně pochopit funkci fragmentů je pro vývoj pokročilejších mobilních aplikací nezbytné. Prostřednictvím této aplikační komponenty lze dynamicky měnit uživatelské rozhraní aplikace a podle dané chvíle tak přizpůsobit obsah, který je nutno zobrazit. Velkou výhodou je jednoduché předávání informací prostřednictvím transakcí. Před samotnou konstrukcí je popsán způsob navigace mezi jednotlivými fragmenty.

Navigace mezi jednotlivými fragmenty

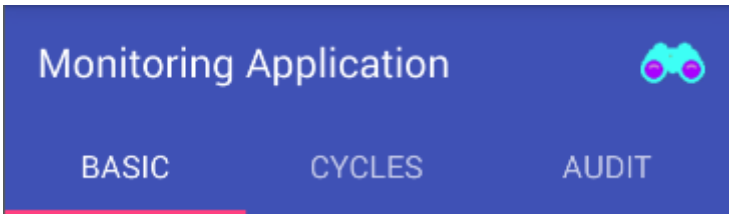
Pro základní ovládání aplikace je vytvořen adaptér umožňující přechod mezi jednotlivými fragmenty. Tento adaptér se nazývá ViewPager. Na obrázku 4-13 je znázorněn princip fungování tohoto adaptéru. Mezi fragmenty se lze pohybovat pomocí tažení obrazovky. S každým fragmentem je asociována pozice sloužící k identifikaci adaptérem (ViewPager). Prostřednictvím tohoto adaptéru lze mezi

jednotlivými fragmenty předávat informace a udržet nepřerušované spojení s pračkou.



Obrázek 4-13 ViewPager pro navigaci mezi fragmenty, zdroj: vlastní

Jako další komponenta, která je využita pro navigaci je TabLayout. Z návrhu uživatele je patrné, jak má přibližně vypadat. Tento layout se běžně využívá pro vytvoření navigační nabídky a skládá se z jednotlivých položek. Tyto položky reprezentují fragmenty vyvíjené aplikace. Na obrázku 4-14 je uveden tento způsob navigace.

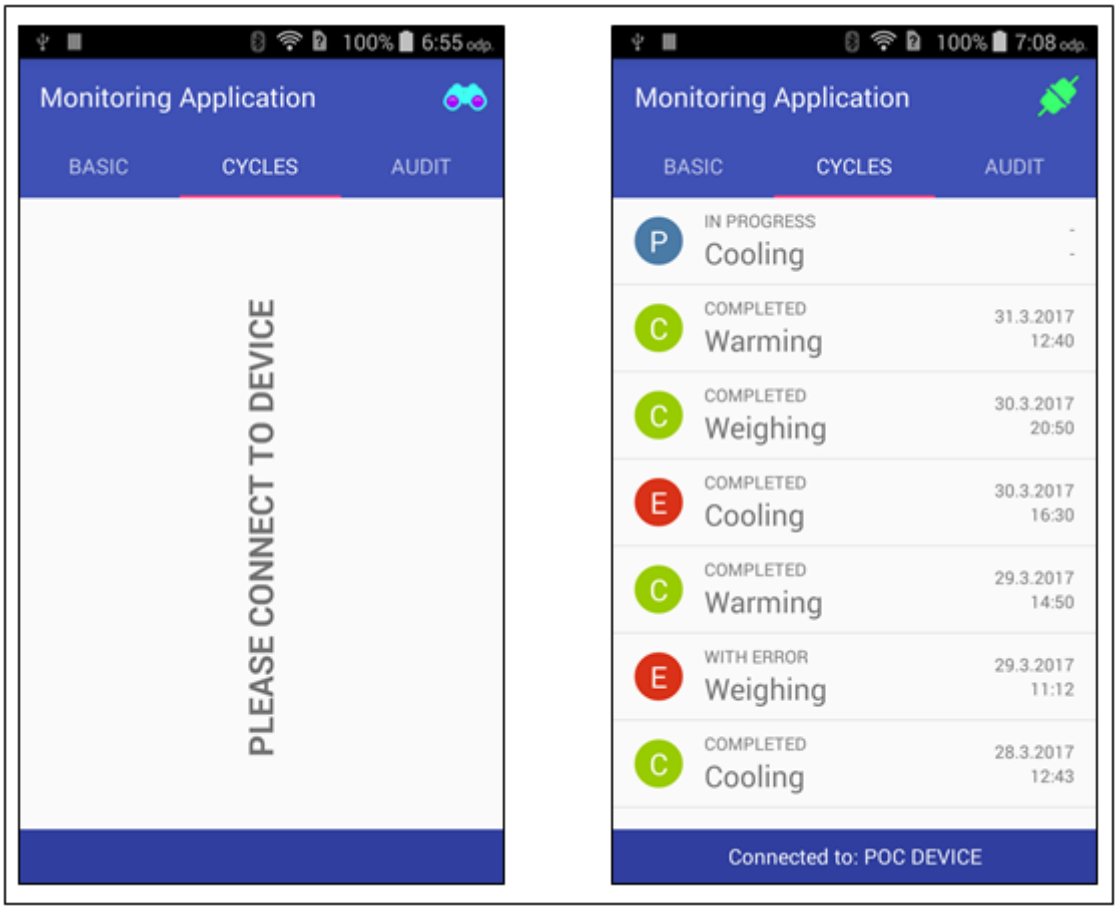


Obrázek 4-14 TabLayout mobilní aplikace, zdroj: vlastní

Vytvoření fragmentu pro obrazovku zobrazující přehled cyklů

Na obrázku 4-15 jsou uvedeny dvě obrazovky, které reprezentuje jeden fragment. V první části se jedná o zobrazení fragmentu, pokud není navázáno spojení s průmyslovou pračkou. V druhé části obrázku již lze vidět zobrazení podrobného přehledu cyklů. Při odpojení nebo posunování obrazovky je nutné jednotlivé cykly recyklovat, aby nedošlo k zahlcení paměti telefonu. Tato recyklace je

implementována v rámci ListView prostřednictvím vytvořeného adaptéru (kapitola 2.3.3).



Obrázek 4-15 Fragment pro zobrazení přehledu cyklů, zdroj: vlastní

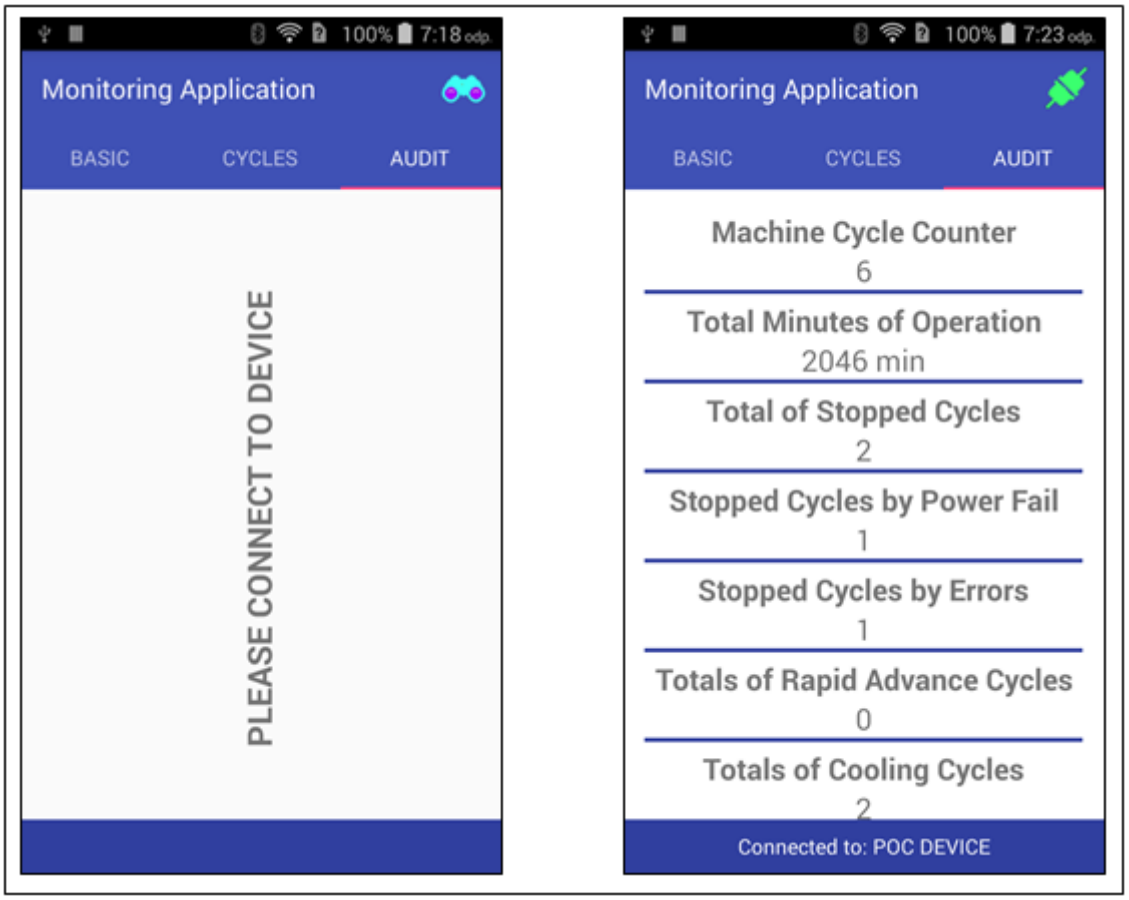
V druhé části obrázku je uveden již samotný přehled cyklů, které proběhly na konkrétní pračce. Název cyklu je uveden s využitím největšího fontu. Jednotlivé cykly jsou seřazeny podle data a času. Každý cyklus obsahuje informaci o svém stavu, který nabývat může jednu z následujících hodnot:

- **P** – cyklus momentálně probíhá,
- **C** – cyklus byl úspěšně dokončen,
- **E** – v průběhu cyklu nastala chyba.

Vytvoření fragmentu pro obrazovku zobrazující auditní informace

Fragment pro zobrazení auditních informací obsahuje jednoduché a přehledné uživatelské rozhraní. Obrázek 4-16 je znovu rozdělen na dvě části, stejně jako v případě předchozího fragmentu. Jednotlivé auditní informace uvedené na obrázku vycházejí z analýzy UX (viz kapitola 4.3.2). Smyslem tohoto fragmentu je

poskytnout uživateli rychlé a přehledné informace, z kterých lze okamžitě dedukovat například chybovost konkrétních cyklů.



Obrázek 4-16 Fragment pro zobrazení přehledu cyklů, zdroj: vlastní

4.3.5 Validace

Mobilní aplikace je úspěšně rozšířena o dvě další obrazovky (fragmenty), které výrazně rozšiřují funkcionalitu. Aplikace je připravena pro předání zadavateli.

Zadavatel je spokojen s uživatelským rozhraním obrazovky (fragmentu) obsahující základní přehled cyklů, nicméně je nutné výrazně rozšířit funkcionality tohoto fragmentu. Jedná se o detailní rozpracování jednotlivých cyklů. Rozšíření o tuto funkcionalitu je náplní příští iterace.

Obrazovka zobrazující auditní informace je hodnocena pozitivně, avšak je nutno provést několik kosmetických úprav (změny velikosti písma apod.). Jedná se opravdu o triviální změny, které již nejsou v práci dále rozebírány. Druhý prototyp vyvíjené mobilní aplikace je úspěšně předán zadavateli a je možné pokračovat v dalším vývoji.

4.4 Vytvoření beta testovací verze aplikace

Při předchozí validaci zadavatel stručně nastínil nové požadavky, které se týkají rozšíření funkcionality fragmentu obsahující přehled jednotlivých cyklů. Náplní této iterace je tedy rozšíření funkcionality tohoto fragmentu. Po dokončení této iterace vývoje je aplikace připravena k nasazení do provozu ve své beta verzi. Pokud se tato mobilní aplikace v praxi osvědčí a ukáže jako přínosná, tak bude dále vyvíjena.

4.4.1 Specifikace požadavků pro beta verzi aplikace

Jak již bylo řečeno, jedná se o rozšíření fragmentu přehledu cyklů. Při kliknutí na daný cyklus (položku ListView) si může uživatel prohlédnout detailní informace.

Jestliže je stav cyklu označen jako **C** (viz kapitola 4.3.4), jsou uživateli poskytnuty detailní informace o průběhu cyklu.

Pokud je stav cyklu označen jako **E** (kapitola 4.3.4), je uživateli zobrazen popis chyby, který v průběhu cyklu nastal. Tento popis lze přeposlat svému nadřízenému.

Poslední možný stav je označen písmenem **P** (kapitola 4.3.4). Jestliže uživatel zvolí tento cyklus, je prostřednictvím vysílací zprávy (kapitola 2.3.3) mobilní aplikace informován o probíhajícím stavu cyklu a času zbývajícího k jeho dokončení.

Podrobný přehled požadavků včetně jejich analýzy je uveden v tabulce 4-4.

Tabulka 4-4 Přehled požadavků pro beta verzi mobilní aplikace, zdroj: vlastní

Tabulka požadavků		
Požadavek	**Analýza**	**Výstup**
Při kliknutí na vybraný cyklus, který je označen jako kompletně dokončený (**C**), se zobrazí detailní informace.	Mobilní aplikaci je nutno rozšířit o další fragment, jímž lze zobrazit informace o kompletně dokončeném cyklu. Nutností je implementovat způsob navigace pro nově vytvořený fragment.	Fragment, který zobrazí korektní informace pro vybraný cyklus.

Při kliknuti na vybraný cyklus, který je označen jako nedokončený s chybovým stavem (**E**) se zobrazí podrobný popis chyby s možnosti rychlého nahlášení nadřízenému.	Rozšíření aplikace o nový fragment obsahující informace o chybovém stavu vybraného cyklu. Důležitá je také interpretace jednotlivých chybových stavů, která je řešena na úrovni fragmentu.	Fragment zobrazující detailní popis chybových stavu jednotlivých cyklů s možnosti nahlášení nadřízenému.
Při kliknutí na cyklus, který právě probíhá (**P**) se zobrazí zpráva oznamující uživateli tento stav a také čas zbývající k dokončení cyklu.	Při kliknutí na probíhající cyklus je uživateli zobrazena zpráva informující o tomto stavu. Zpráva obsahuje také čas zbývající k dokončení cyklu.	Jedná se o modifikaci již existujícího fragmentu cyklů.
Přeložit aplikaci do češtiny.	Využití univerzálních textových řetězců, které se dají následně překládat do jakéhokoliv jazyku.	Výstupem je aplikace v češtině a angličtině.

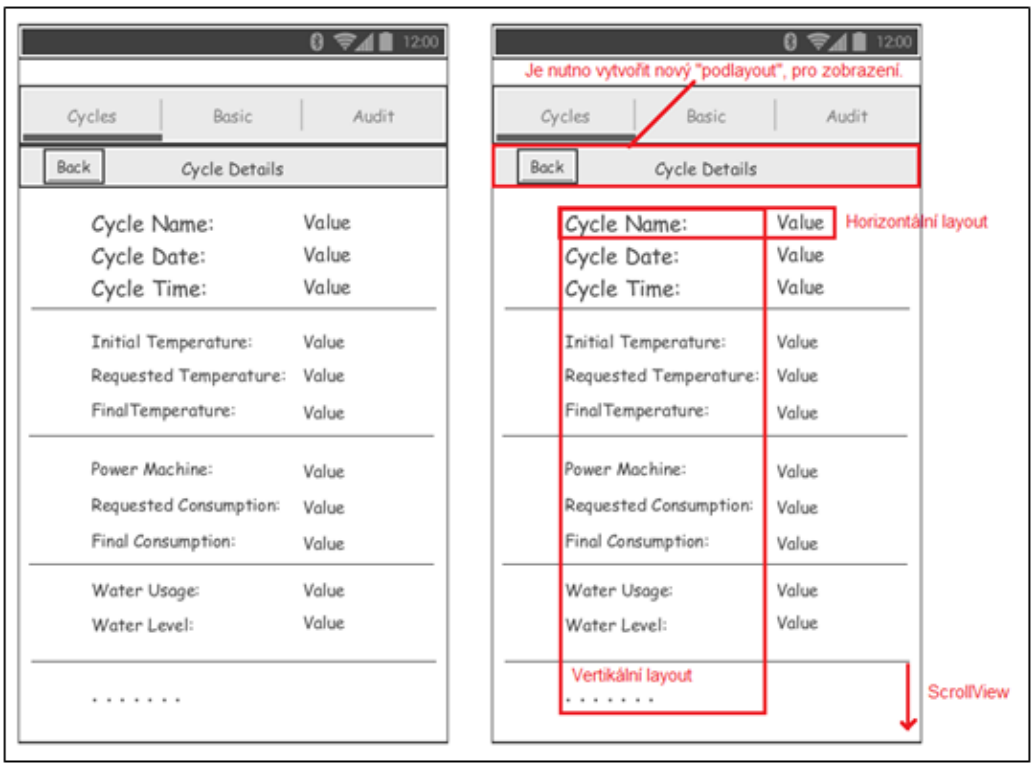
4.4.2 Návrh beta verze aplikace s využitím UX

V této kapitole je popsán návrh obrazovek (fragmentů) reprezentující výše uvedené požadavky (kapitola 4.4.1). Jako v celé práci je i zde využito sady technik UX.

V první části této kapitoly je uveden uživatelův návrh obrazovky s detailními informacemi o kompletně dokončeném cyklu. Dále je popsána obrazovka pro detailní zobrazení chybového cyklu. V poslední části je popsán návrh úpravy již stávajícího fragmentu při kliknutí na probíhající cyklus.

Návrh obrazovky s detailními informacemi o dokončených cyklech

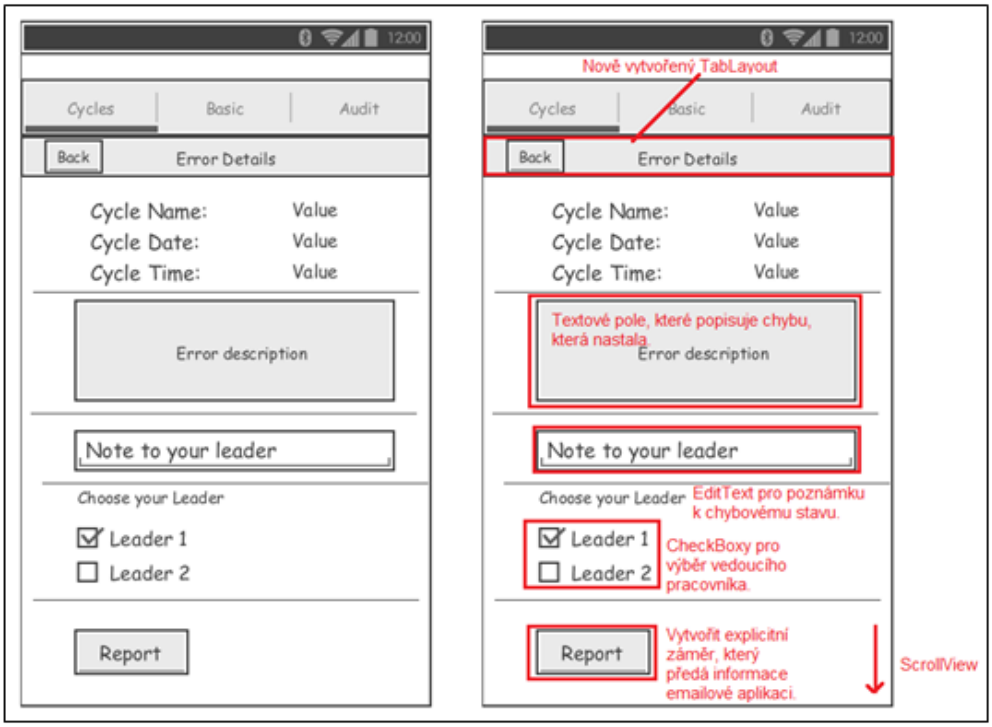
Na obrázku 4-17 je uveden uživatelův návrh obrazovky včetně stručné analýzy vývojáře. Ačkoliv uživatel nastínil, jaké informace tento fragment musí obsahovat, v praktickém provedení je toto zadání daleko rozsáhlejší, jelikož pro každý typ cyklu se vypisují specifické informace. Z analýzy obrázku vyplývá, že je nutno vytvořit TabLayout zobrazující název nově vytvořeného fragmentu včetně možnosti návratu zpět prostřednictvím tlačítka. Jelikož tato obrazovka (fragment) obsahuje velké množství údajů, je nutné znovu využít ScrollView pro implementaci posunovací funkce. Uživatelské rozhraní je tvořeno kombinací obou typů lineárního layoutu (kapitola 2.3.3).



Obrázek 4-17 Návrh obrazovky pro zobrazení detailního přehledu dokončených cyklů, zdroj: vlastní

Návrh obrazovky s detailním popisem chyby nedokončeného cyklu

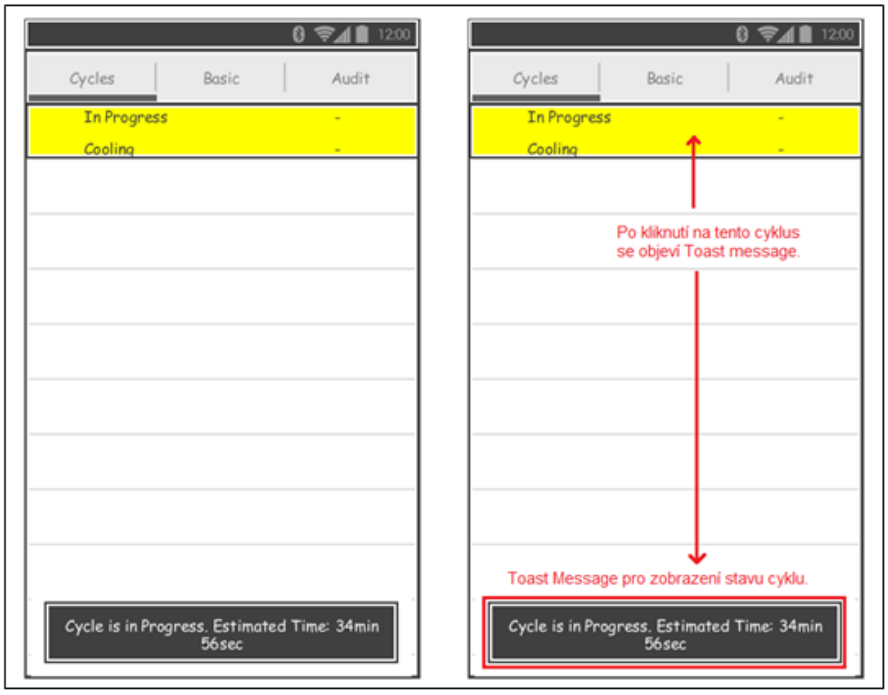
Na obrázku 4-18 je uveden uživatelův návrh obrazovky pro detailní popis chyby včetně stručné analýzy vývojáře, která nastala během cyklu. Kromě základních informací o nedokončeném cyklu a již zmíněného TabLayoutu obsahuje obrazovka textové pole s detailním popisem chyby. Pro uživatele je zde k dispozici možnost uvést konkrétní poznámku k chybovému stavu. Tento popis chyby lze odeslat konkrétnímu vedoucímu pracovníkovi prostřednictvím zaškrtávacích políček reprezentující jednotlivé nadřízené. Pro samotné odeslání slouží tlačítko na spodní části obrazovky, jehož úkolem je vytvořit explicitní záměr (kapitola 2.3.3) a požadované informace tak předat poštovní aplikaci.



Obrázek 4-18 Návrh obrazovky pro zobrazení detailních informacích o dokončeném cyklu, zdroj: vlastní

Návrh pro úpravu obrazovky při výběru momentálně probíhajícího cyklu

Na obrázku 4-19 je zobrazena zpráva, která se uživateli zobrazí při kliknutí na cyklus, který momentálně probíhá. Zpráva obsahuje oznámení o stavu cyklu včetně doby potřebné k jeho dokončení. Jedná se o úpravu již existujícího fragmentu.



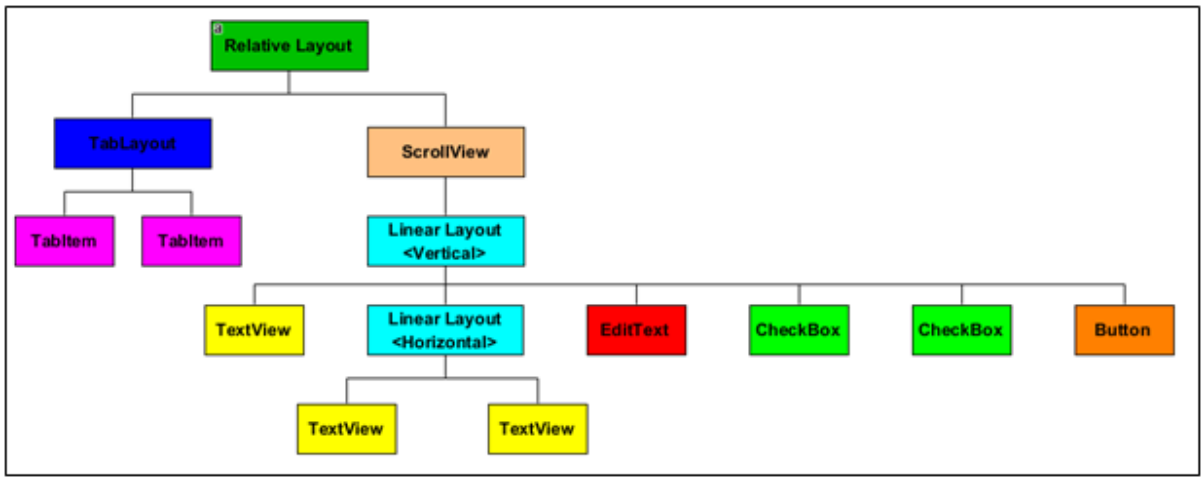
Obrázek 4-19 Návrh úpravy obrazovky obsahující přehled cyklů při kliknutí na probíhající cyklus, zdroj: vlastní

4.4.3 Návrh layoutu UI s využitím hierarchie pohledů

V této kapitole je uveden diagram hierarchie pohledů pouze pro obrazovku (fragment) obsahující detailní popis chyby cyklu. Tento diagram není nutné vytvářet pro obrazovku s detailními informacemi o kompletně dokončeném cyklu, jelikož se jedná o analogický postup k již první vytvořené obrazovce se základními informacemi o pračce (viz kapitola 4.2.3).

Diagram hierarchie pohledů pro obrazovku s detailním popisem chyby

Na obrázku 4-20 je uveden diagram hierarchie pohledů, který reprezentuje layout uživatelského rozhraní pro danou obrazovku. Na vrcholu hierarchie je umístěn relativní layout, s jehož využitím lze rozmístit jednotlivé pohledy (Views) k dosažení požadovaného designu. TabLayout reprezentuje vytvoření nové hlavičky pro tento fragment. Druhým přímým potomkem relativního layoutu je ScrollView, které obaluje veškerou obsahovou část tohoto uživatelského rozhraní s možností posunování obrazovky. Skladba obsahové části obrazovky je patrná z obrázku.



Obrázek 4-20 Diagram hierarchie pohledu pro obrazovku s detailním popisem chyby cyklu, zdroj: vlastní

4.4.4 Vytvoření fragmentů pro rozšířený popis jednotlivých cyklů

Tato kapitola je věnována samotné konstrukci jednotlivých fragmentů. V první části je popsán nový způsob navigace a komunikace mezi nově vzniklými obrazovkami. V další části jsou detailně rozebrány jednotlivé fragmenty.

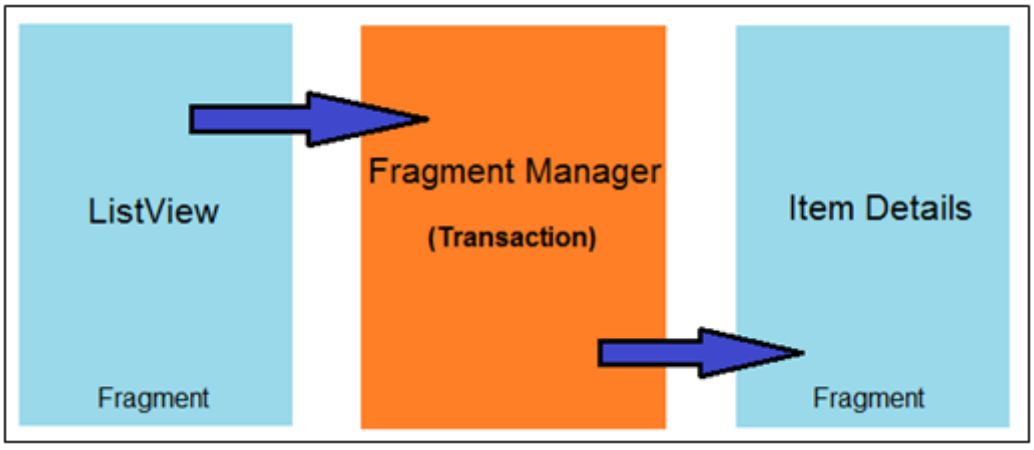
Návrh nového způsobu navigace a komunikace mezi fragmenty

Pro přechod na nové fragmenty obsahující informace o konkrétních cyklech nelze využit již existující ViewPager adaptér (kapitola 4.3.4). Je tedy nutné implementovat způsob, který umožní plynulý přechod mezi fragmentem obsahujícím základní přehled cyklů a fragmenty, kde jsou tyto cykly detailně popsány.

Pro komunikaci a přechod mezi jednotlivými fragmenty lze využit transakce, prostřednictvím kterých lze předávat důležité informace a dynamicky měnit uživatelské rozhraní. Jako transakci mezi fragmenty lze označit posloupnost kroků, která vede k:

- přidání nového fragmentu,
- odebrání již existujícího fragmentu,
- nahrazení stávajícího fragmentu.

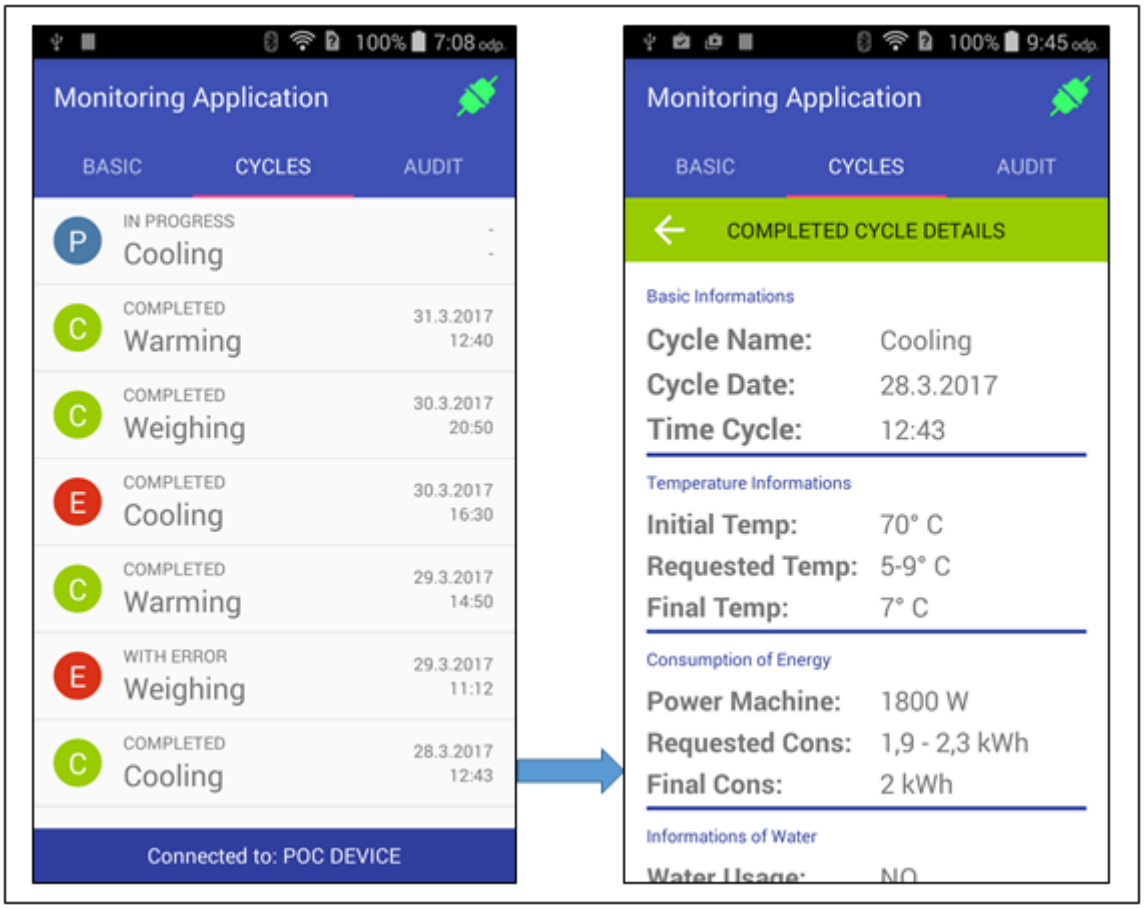
Na obrázku 4-21 je graficky znázorněn přechod mezi dvěma fragmenty s využitím transakcí. Po kliknutí na konkrétní cyklus v rámci ListView je provedena transakce, která je implementována v rámci třídy Fragment Manager. Transakce zahrnuje posloupnost kroků, která se musí provést celá. Cílem je přechod na konkrétní fragment, v rámci kterého jsou zobrazeny informace o vybraném cyklu. Prostřednictvím transakcí si lze předávat důležité informace, které mohou sloužit například k identifikaci konkrétních cyklů.



Obrázek 4-21 Transakce mezi fragmenty, zdroj: vlastní

Vytvoření fragmentu pro detailní popis dokončeného cyklu

Na obrázku 4-22 je uveden již vytvořený fragment pro detailní popis úspěšně dokončených cyklů. Po kliknutí na konkrétní cyklus je uživatel prostřednictvím transakcí přesměrován na nově vzniklý fragment. Fragment obsahuje jednoduché a přehledné uživatelské rozhraní, které detailně popisuje vybraný cyklus. Identifikace jednotlivých cyklů je prováděna prostřednictvím identifikačních čísel, které jsou s jednotlivými cykly asociovány. Tyto identifikační čísla si jednotlivé fragmenty prostřednictvím transakcí předávají a na jejichž základě je dynamicky zobrazen obsah obrazovky.

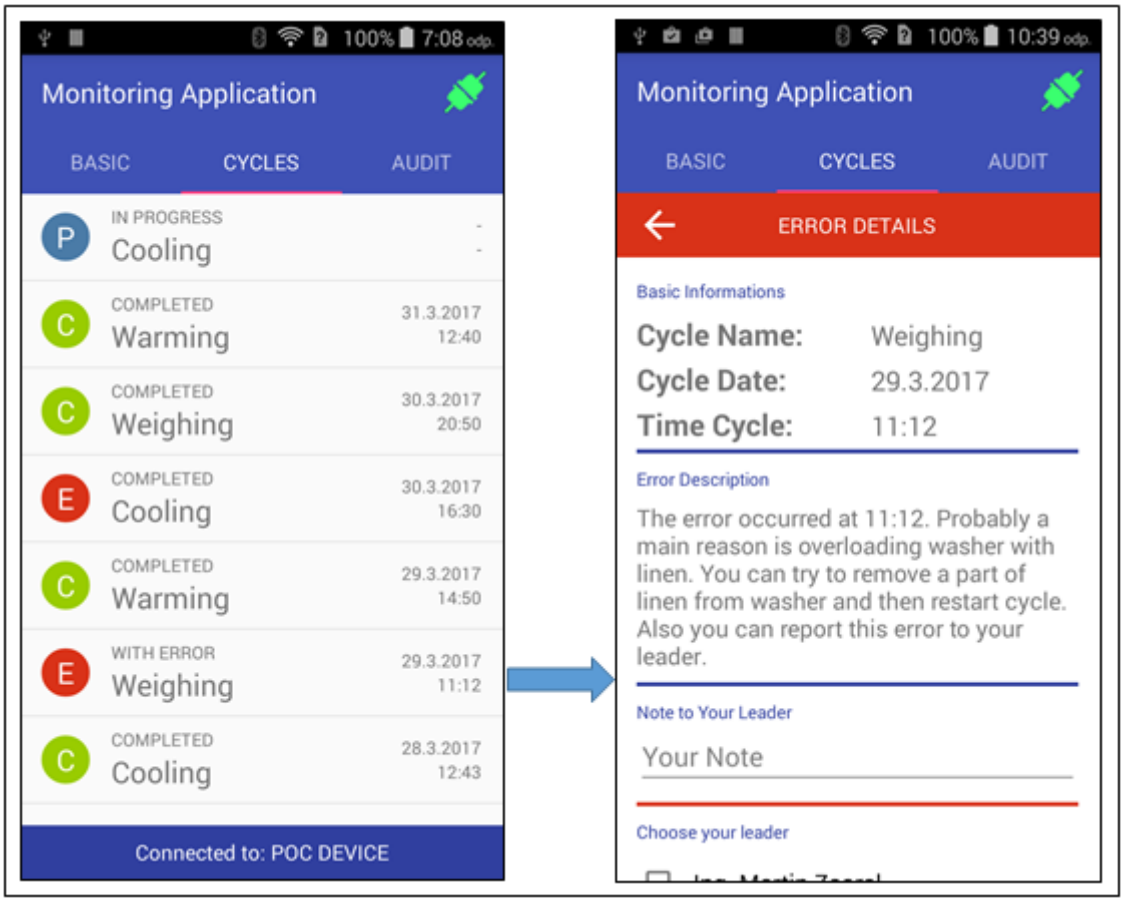


Obrázek 4-22 Fragment obsahující detailní popis úspěšně dokončených cyklů, zdroj: vlastní

Vytvořený fragment obsahuje více informací, ale pro ilustrativní účely je uvedený obrázek dostačující. Pro návrat zpět k zobrazení přehledu cyklů lze využít šipku, která je umístěna v nově vytvořeném TabLayoutu. Jelikož se při transakci původní fragment ukládá do zpětného zásobníku, pro přechod zpět není nutné vyvolávat novou transakci.

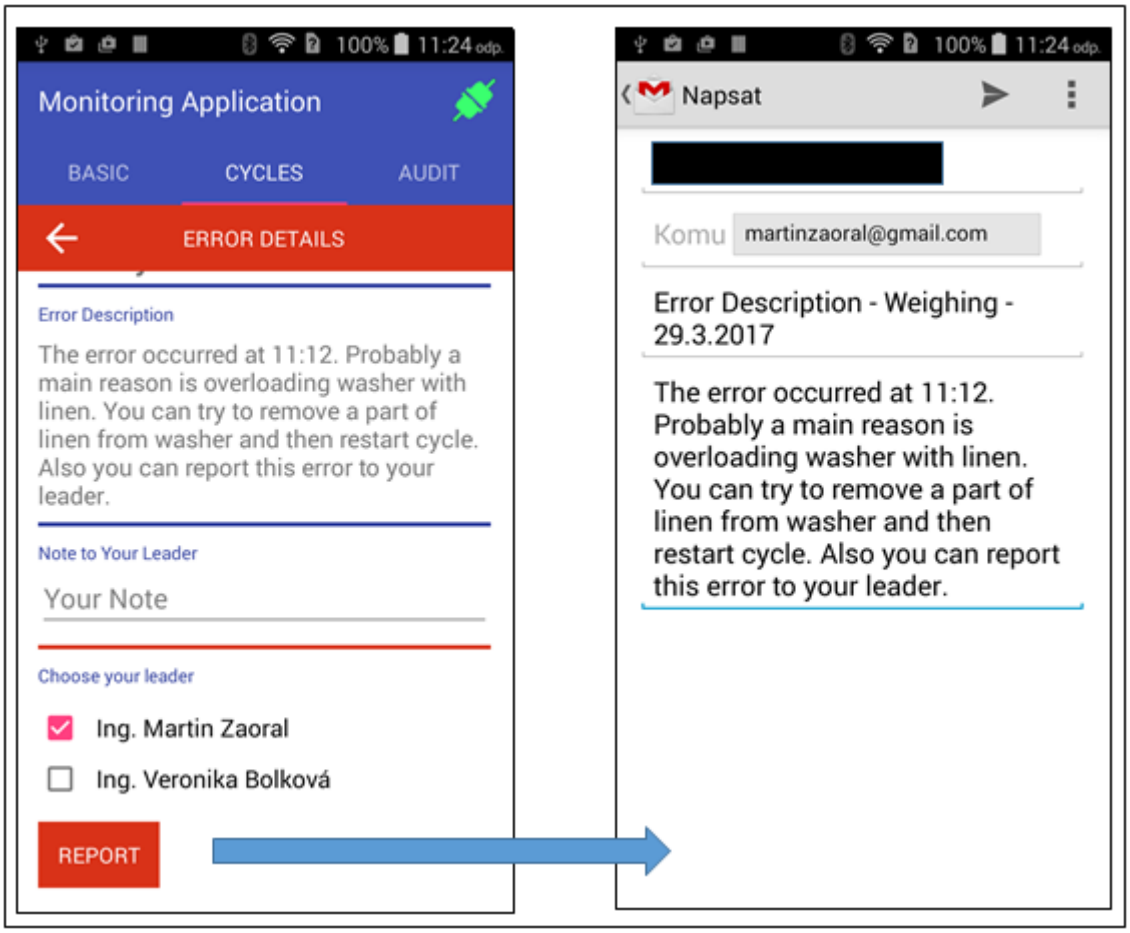
Vytvoření fragmentu pro popis chyby nedokončeného cyklu

Na obrázku 4-23 je uvedený fragment pro detailní popis chyby nedokončeného cyklu. V první obsahové části fragmentu jsou uvedeny základní informace o cyklu včetně času, kdy nastala chyba. V druhé části je uvedena samotná interpretace chyby, která se provádí na úrovni tohoto fragmentu. K popisu chyby je možné doplnit komentář prostřednictvím textového editačního pole. Ten poté lze odeslat vybranému vedoucímu pracovníkovi.



Obrázek 4-23 Fragment obsahující detailní popis chyby, zdroj: vlastní

Druhá polovina fragmentu je uvedena na obrázku 4-24 včetně možnosti nahlášení chybového stavu. Pomocí zaškrtávacích políček lze vybrat konkrétního vedoucího pracovníka a chybu mu odeslat. Pro odeslání chybových informací je využit explicitní záměr (viz kapitola 2.3.3), kdy jsou jiné mobilní aplikaci v telefonu předány specifické informace. Pro účely této práce je vybrána aplikace Gmail od společnosti Google. V praxi je však využívána interní poštovní aplikace. Z druhé části obrázku je patrné, jak může obsah dynamicky vytvořené emailové zprávy vypadat.



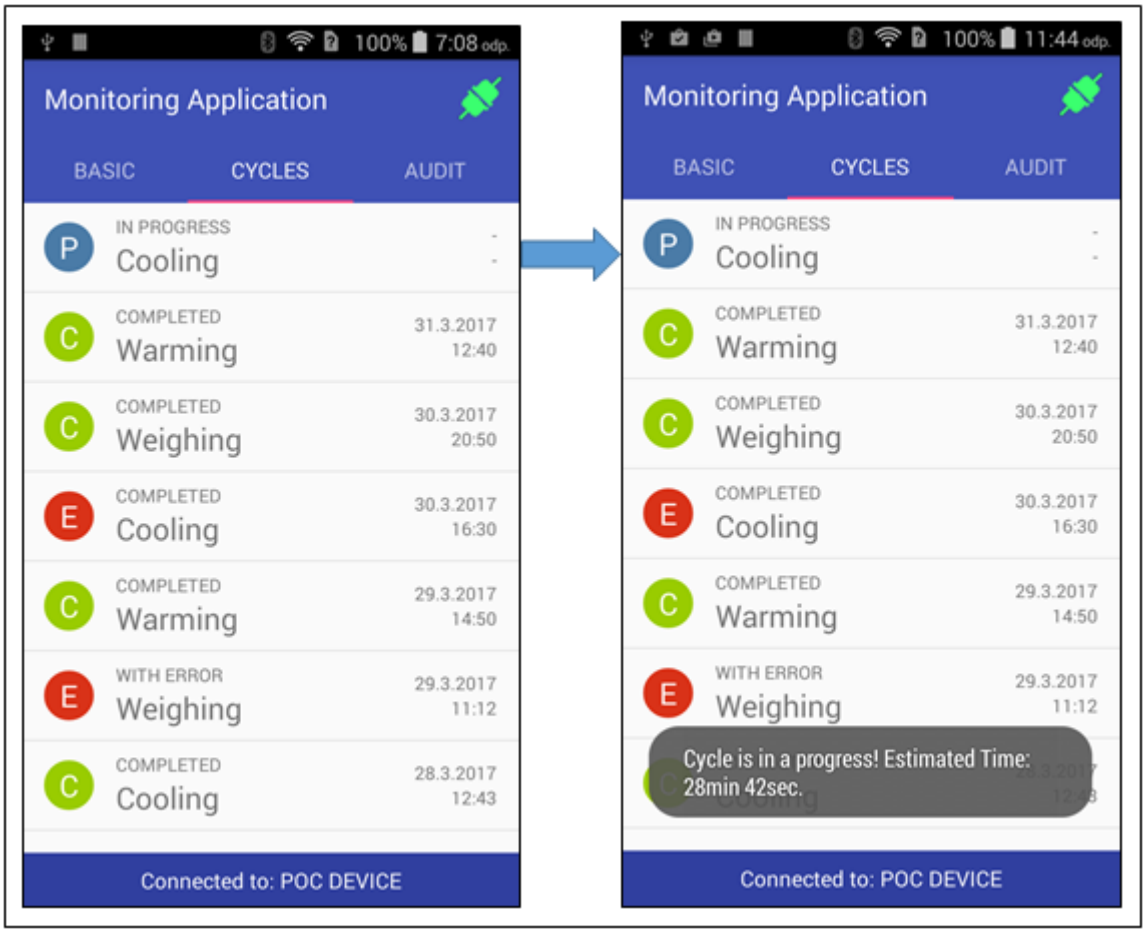
Obrázek 4-24 Fragment obsahující popis chybového cyklu včetně možnosti nahlášení, zdroj: vlastní

Pro zachování integrity jsou informace uvedené na obrázku pochopitelně fiktivního charakteru.

Upravení již existujícího fragmentu přehledu cyklů

Poslední částí v rámci této podkapitoly je upravení již stávajícího fragmentu, prostřednictvím kterého je zobrazen přehled cyklů.

Na obrázku 4-25 je zobrazen původní fragment přehledu cyklů. Při kliknutí na cyklus, který momentálně probíhá je uživateli zobrazena stavová zpráva. Obsahem zprávy je informace o probíhajícím stavu a současně čas potřebný pro dokončení cyklu. Dokud se cyklus nachází ve stavu **P** (kapitola 4.3.4), nelze zobrazit detailní informace. Při kliknutí na tento cyklus je vyvolán dotaz, prostřednictvím kterého mobilní aplikace zjišťuje čas potřebný pro dokončení cyklu. Komunikace tedy probíhá v reálném čase.



Obrázek 4-25 Rozšíření fragmentu zobrazující přehled cyklů o novou funkcionalitu, zdroj: vlastní

4.4.5 Lokalizovatelnost aplikace pro podporu více jazyků

Android běží na mnoha zařízeních v nejrůznějších regionech. K dosažení toho, aby mobilní aplikace byla co nejvíce využitelná, je nutné spravovat text, obrázky, čísla, grafický design a mnoho dalšího.

Předmětem této kapitoly je pro vyvíjenou aplikaci implementovat podporu pro Český jazyk. Mobilní aplikace může obsahovat zdroje, které jsou specifické pro jednotlivé národnosti. Jedná se například o využití specifických textových řetězců, které jsou přeloženy do vybraného jazyka podle současné lokalizace. Následné jazykové nastavení je pak vybráno podle lokálního nastavení uživatelova telefonu.

V tabulce 4-5 je uvedena část specifických textových řetězců, prostřednictvím kterých lze přeložit původně navržený anglický text do českého překladu. Z tabulky je intuitivně jasné, jak celý tento proces probíhá. Ve zdrojovém souboru se definují klíčové výrazy, včetně jejich interpretace v Anglickém jazyce. Následně se takto

definované textové řetězce využívají napříč celou mobilní aplikací a lze je přeložit do libovolného jazyka.

Tabulka 4-5 Ukázka textových řetězců určených pro lokalizaci, zdroj: vlastní

Key	Untranslata...	Default Value	Czech (cs)
app_name		Monitoring Application	Aplikace pro monitoring
audit		Audit	Audit
basic		Basic	Základní
connect_button		Connect	Připojit
controller_type		Controller Type	Typ ovladače
cycle_name		Cycle Name	Název cyklu
cycles		Cycles	Cykly
disconnect_button		Disconnect	Odpojit
install_date		Installation Date	Datum instalace
machine_status		Machine Status	Stav stroje
menu_connect		Connect	Připojit
menu_disconnect		Disconnect	Odpojit
menu_exit		Exit	Odejít
model_number		Model Number	Číslo modelu
sw_group		SW Group	SW Skupina
sw_version		SW Version	SW Verze
temperature		Temperature	Teplota

4.4.6 Validace

Jednotlivé požadavky jsou úspěšně převedeny do funkcionalit mobilní aplikace. Před samotným nasazením mobilní aplikace je nutné provést poslední validaci.

Zadavatel je s implementací jednotlivých funkcionalit spokojen. Před nasazením aplikace do testovacího provozu je však nutné provést ještě malé změny u jednotlivých fragmentů. Pro fragment obsahující detailní informace o cyklech je nutné přesně specifikovat obsah informací. Pro druhý vytvořený fragment v rámci této iterace je potřeba lehce pozměnit strukturu implementace chybových stavů a samozřejmě fiktivní údaje nahradit reálnými. Tyto změny již nejsou součástí této práce, jelikož se jedná o triviální úpravy.

4.4.7 Nasazení aplikace a návrh budoucího vývoje

V tuto chvíli je aplikace připravena k prvnímu nasazení do provozu. Testovací období bude probíhat zhruba jeden měsíc. Během tohoto testování budou uživatele poskytovat zpětnou vazbu, která povede k detailnějšímu doladění a další modifikaci.

Před samotným nasazením aplikace je vhodné provést krátké školení, v rámci kterého bude aplikace představena a vysvětlena.

Distribuce vytvořené mobilní aplikace je zprostředkována pomocí e-mailové komunikace. Jedná se o rychlou a jednoduchou cestu. Více o tomto způsobu distribuce aplikace je uvedeno v teoretické části práce (kapitola 2.3.6).

Po nasazení mobilní aplikace do testovací fáze je potřeba pokračovat v plánování vývoje. Jedná se teprve o beta verzi aplikace, která obsahuje jenom základní funkcionality, které je nutno dále rozšiřovat.

Ve firmě právě probíhá vývoj, jehož předmětem je implementace síťové komunikace pro pračky. Jakmile bude tento projekt dokončen, lze uživatelské rozhraní aplikace využít i pro klasické síťové připojení, nikoliv jen pro Bluetooth.

Aktuálně jsou jednotlivé návrhy pro budoucí vývoj následující:

- rozšíření funkcionality aplikace o možnost exportu dat na interní úložiště,
- implementace základního způsobu ovládání pračky (zamykání dveří, manuální pozastavení cyklů apod.).

5 Závěr

Cílem diplomové práce bylo navrhnout a vytvořit servisní mobilní aplikaci, která slouží pro monitorování průmyslových praček. V rámci této diplomové práce bylo nutné získat znalosti z oblasti vývoje softwaru do těchto zařízení. Dále pak pochopit základní principy komunikace prostřednictvím technologie Bluetooth. Klíčovým pro dosažení cílů bylo však nabytí znalostí nutných pro vývoj mobilních aplikací na platformě Android. Pro praktické vytvoření mobilní aplikace byly tyto nabyté znalosti detailně využity. Celý vývojový proces byl řízen prostřednictvím metodiky UP a prvky agilního vývoje.

V kapitole teoretická východiska tvorby mobilních aplikací v oblasti průmyslových praček byly popsány koncepty, které jsou pro návrh a implementaci vyvíjené mobilní aplikace využity. Tato kapitola obsahuje charakteristiku technologií průmyslových praček včetně možnosti jejich komunikace s mobilním zařízením. Dále pak popis metodiky pro vývoj softwaru včetně prvků agilního vývoje. Hlavní části této kapitoly je popis mobilního operačního systému Android a to převážně z pohledu vývojáře. Na závěr jsou uvedeny základní technologie, které byly využity.

Druhou kapitolou práce je analýza současného stavu podniku v oblasti komunikace. V této kapitole byly uvedeny informace o společnosti včetně popisu technologických možností firmou vyvíjených průmyslových praček v oblasti komunikace. Dále byl v této kapitole popsán současný stav firmy v oblasti monitoringu průmyslových praček.

V kapitole návrh a implementace vyvíjené aplikace byl popsán celý vývojový proces servisní mobilní aplikace zahrnující návrh, implementaci a nasazení. Vývoj je iterativní a řízený metodikou UP. Zařazeny jsou i prvky agilního vývoje, jako například sada technik UX nebo pravidelné konzultace se zadavatelem. V první iteraci byly sesbírány základní požadavky na mobilní aplikaci. Výstupem této iterace byla analýza jednotlivých požadavků s využitím grafické anotace UML. Obsahem druhé iterace bylo praktické navržení a implementace prvního prototypu vyvíjené aplikace. Tento prototyp sloužil také jako základní návrh architektury pro další vývoj a implementoval komunikaci průmyslové pračky s mobilním zařízením prostřednictvím technologie Bluetooth včetně úvodní obrazovky mobilní aplikace. Ve třetí iteraci byly již detailněji specifikovány požadavky, na jejichž základě byla rozšířena funkcionalita

mobilní aplikace například v podobě zobrazení jednotlivých cyklů nebo auditních informací o pračce. Poslední iterace vývoje zahrnovala detailnější rozšíření různých funkcionalit na základě znovu specifikovaných požadavků. V této iterace byla mobilní aplikace uvedena do testovací verze beta a nasazena k provozu. Dále byly uvedeny návrh na možný budoucí vývoj mobilní aplikace.

Po dokončení této diplomové práce lze konstatovat, že cíle stanovené v úvodu byly splněny. Mobilní aplikace má do budoucna velký potenciál, jelikož v současné době je proces, zabývající se monitoringem průmyslových praček neefektivní. Při využívání servisní mobilní aplikace a jejím dalším vývoji lze očekávat výraznou optimalizaci tohoto procesu.

Samotný vývoj mobilních aplikací v rámci společnosti Alliance Laundry CE, nabízí v současné době mnoho možností pro konstrukci nových mobilních aplikací. To je samozřejmě spojené se současnými trendy a internetem věcí. Lze tak vytvářet mobilní aplikace různých charakterů, například pro platební styk s pračkou v prádelnách, nebo vytvoření vlastního uživatelského rozhraní, které bude ovládáno prostřednictvím mobilního telefonu.

Seznam použité literatury

Odborné knihy

ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press, 2007. 567 s. ISBN 978-80-251-1503-9.

BUTTAZZO, Giorgio C. *Soft real-time systems: predictability vs. efficiency*. New York: Springer, 2005. Series in computer science (New York, N.Y.). ISBN 978-0-387-23701-5.

BUTTAZZO, Giorgio C. *Hard real-time computing systems: predictable scheduling algorithms and applications*. 3rd ed. New York: Springer, c2011. Real-time systems series (Springer). ISBN 978-1-4614-0675-4.

DOUGLASS, Bruce Powel. *Doing hard time: developing real-time systems with UML, objects, frameworks, and patterns*. Boston: Addison-Wesley, c1999. ISBN 978-0321774934.

HAMILTON, Kim and Russell MILES. *Learning UML 2.0*. Sebastopol: O'Reilly Media, 2006. 286 p. ISBN: 978-0-59-600982-3.

KADLEC, Václav. *Agilní programování: metodiky efektivního vývoje softwaru*. Brno: Computer Press, 2004. ISBN 80-251-0342-0.

KOPETZ, Hermann. *Real-time systems: design principles for distributed embedded applications*. 2nd ed. Boston: Springer, c2011. Real-time systems series. ISBN 978-1-4419-8236-0.

LACKO, Ľuboslav. *Vývoj aplikací pro Android*. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

OSHANA, Robert and Mark KRAELING. *Software engineering for embedded systems: methods, practical techniques, and applications*. Boston: Elsevier/Newnes, 2013 ISBN 978-0124159174.

POHL, K., M. BROY, H. DAEMBKES, H. HÖNNIGER. *Advanced model-based engineering of embedded systems*. Cham: Springer International Publishing AG, 2016. ISBN 978-3-319-48002-2.

SMITH, Dave. *Android recipes: a problem-solution approach for Android 5.0*. Fourth ed., New York: Apress, 2015. ISBN 978-1-484204-76.

STOBER, Thomas and Uwe HANSMANN. *Agile software development: best practices for large software development projects*. New York: Springer, c2010. ISBN 978-3-540-70830-8.

Elektronické dokumenty a ostatní

Android Developers. *Bluetooth Low Energy* [online]. c2017a [cit. 2017-04-16]. Dostupné z: https://developer.android.com/guide/topics/connectivity/Bluetooth-le.html

Android Developers. *Android, the world's most popular mobile platform* [online]. c2017b [cit. 2017-04-16]. Dostupné z: https://developer.android.com/about/android.html

Android Developers. *Platform Architecture* [online]. c2017c [cit. 2017-04-16]. Dostupné z: https://developer.android.com/guide/platform/index.html

Android Developers. *Application Fundamentals* [online]. c2017d [cit. 2017-04-16]. Dostupné z: https://developer.android.com/guide/components/fundamentals.html

Android Developers. *Intents and Intent Filters* [online]. c2017e [cit. 2017-04-16]. Dostupné z: https://developer.android.com/guide/components/intents-filters.html

Android Developers. *UI Overview* [online]. c2017f [cit. 2017-04-16]. Dostupné z: https://developer.android.com/guide/topics/ui/overview.html

Android Developers. *Layouts* [online]. c2017g [cit. 2017-04-16]. Dostupné z: https://developer.android.com/guide/topics/ui/declaring-layout.html

Android Developers. *Fragments* [online]. c2017h [cit. 2017-04-16]. Dostupné z: https://developer.android.com/guide/components/fragments.html

Android Developers. *Dashboards* [online]. c2017i [cit. 2017-04-16]. Dostupné z: https://developer.android.com/about/dashboards/index.html

Android Developers. *Alternative distribution options* [online]. c2017j [cit. 2017-04-16]. Dostupné:https://developer.android.com/distribute/marketing-tools/alternative-distribution.html

Android Studio. Meet Android Studio [online]. c2017 [cit. 2017-04-17]. Dostupné z: https://developer.android.com/studio/intro/index.html

Bluetooth SIG, Inc. *What is Bluetooth?* [online]. c2017 [cit. 2017-04-16]. Dostupné z: https://www.Bluetooth.com/what-is-Bluetooth-technology/how-it-works

ČECHÁK, Martin. *Návrh a implementace IS pro cateringovou společnost. Ostrava, 2015*. Bakalářská práce. Vysoká škola báňská - Technická univerzita Ostrava. Ekonomická fakulta.

MULLIS, Alex. *Best Android developer tools* [online]. c2016 [cit. 2017-04-17]. Dostupné z: http://www.androidauthority.com/best-android-developer-tools-671650/

NORMAN, Don a Jakob NIELSEN. The Definition of User Experience (UX) [online]. c1998-2017 [cit.2017-04-16].Dostupné z: https://www.nngroup.com/articles/definition-user-experience/

Techopedia. *Java Development Kit (JDK)* [online]. c2015 [cit. 2017-04-17]. Dostupné z: https://www.techopedia.com/definition/5594/java-development-kit-jdk

Seznam zkratek

ADT Android Developer Tools

AOT Ahead-of-Time

API Application Programming Interface

APK Android Package

ART Android Runtime

AVD Android Virtual Device

BLE Bluetooth Low Energy

CSV Comma Separated Value

DVM Dalvik Virtual Machine

GATT Generec Attribute

GUI Graphical User Interface

HAL Hardware Abstract Layer

IDE Integrated Development Enviromemnt

JAR Java Archive

JDK Java Development Kit

JIT Just-in-Time

JRE Java Runtime Enviroment

JVM Java Virtual Machine

OOP Object Oriented Programming

OS Operation Systém

PAN Personal Area Network

RUP Rational Unified Process

SDK Software Development Kit

UI User Interface

UML Unified Modelling Language

UP Unified Proccess

USB Universal Serial Bus

UX User Experience

XML eXtensible Markup Language

Prohlášení o využití výsledků diplomové práce

Prohlašuji, že:

- jsem byl seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užít
  (§ 35 odst. 3);
- souhlasím s tím, že diplomová práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že bibliografické údaje o diplomové práci budou zveřejněny v informačním systému VŠB-TUO;
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;
- bylo sjednáno, že užít své dílo, diplomovou práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

Ve Frýdku - Místku dne 19. 4. 2017

.....

Bc. Aleš Miňo

Seznam příloh

Příloha č. 1: Fotografie průmyslové pračky

Obrázek 1 Fotografie průmyslové pračky, zdroj: vlastní

Příloha č. 2: Ukázka datagramu pro interpretaci získaných dat

Tabulka 1 Ukázka datagramu pro interpretaci získaných dat z pračky, zdroj: vlastní

Status Request		
Field	**Type**	**Name**
Data1	Byte (8 bit)	Machine Status 1
Data2	Byte (8 bit)	Error nr on Display (higher byte)
Data3	Byte (8 bit)	Selected Program nr
Data4	Byte (8 bit)	Block nr in execution
Data5	Unsigned Long (32 bit)	Estimated rest time in seconds
Data6	Byte (8 bit)	Time counting in hold mode
Data7	Byte (8 bit)	Error nr on display (lower byte)
Data8	Unsigned Long (32 bit)	Error counter
Data9	Integer (16 bit)	Software version and controller type
Data10	Byte (8 bit)	Machine type selected
Data11	Byte (8 bit)	Standard mode Selected

Machine Status 1 - interpretation	
Code	**Status**
0	Machine is busy with power-up
1	Machine Free
2	Machine busy with wash
3	Machine in pause mode
4	Machine in stop / continue mode
5	Machine Done and in error mode
6	Machine Done (waiting for door open)
7	Machine in external communication locking (for program data downloading)
8	Machine is in Menu mode
9	Machine is busy with restarting from power-down
10	Machine Free with error